Scientific Illustrations and Vector Graphics with LaTeX

Lib4RI Module 5+ – 2023 – Empa Dübendorf

Théo Cavalieri[1,*] and Frank Hösli[2,†]

[1] Empa, Swiss Federal Laboratories for Materials Science and Research, Überlandstrasse 129, 8600 Dübendorf, Switzerland
[2] Eawag, Swiss Federal Laboratories for Materials Science and Research, Überlandstrasse 129, 8600 Dübendorf, Switzerland
[*] Contact: Theo.Cavalieri@empa.ch – tcaval@univ-lemans.fr
[†] Contact: Frank.Hoesli@lib4ri.ch – Frank.Hoesli@eawag.ch

June 8th, 2023

Lib4RI
Library for the Research Institutes within the ETH Domain: Eawag, Empa, PSI & WSL

Program of the course

Motivation

For young (and even advanced scientists) creating beautiful figures is complicated. Why?

Motivation

For young (and even advanced scientists) creating beautiful figures is complicated. Why?

- All figures are different, it is hard to establish a consistent workflow

Motivation

For young (and even advanced scientists) creating beautiful figures is complicated. Why?

- All figures are different, it is hard to establish a consistent workflow
- Scientific figures are subject to multiple changes before publication

Motivation

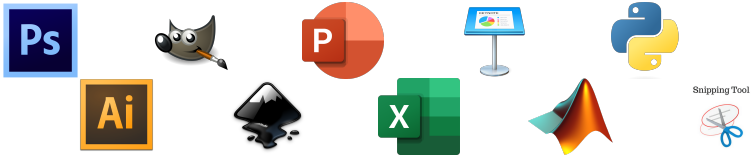
For young (and even advanced scientists) creating beautiful figures is complicated. Why?

- All figures are different, it is hard to establish a consistent workflow
- Scientific figures are subject to multiple changes before publication
- They often rely on multiple processes over different software

Motivation

For young (and even advanced scientists) creating beautiful figures is complicated. Why?

- All figures are different, it is hard to establish a consistent workflow
- Scientific figures are subject to multiple changes before publication
- They often rely on multiple processes over different software



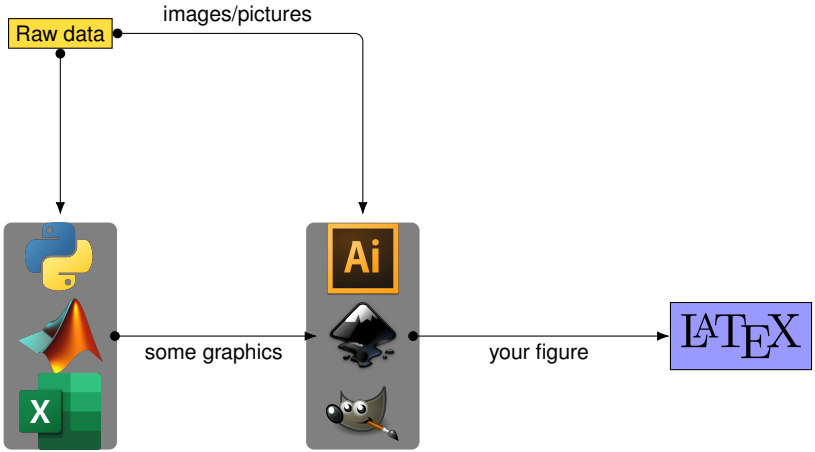
Motivation

For young (and even advanced scientists) creating beautiful figures is complicated. Why?

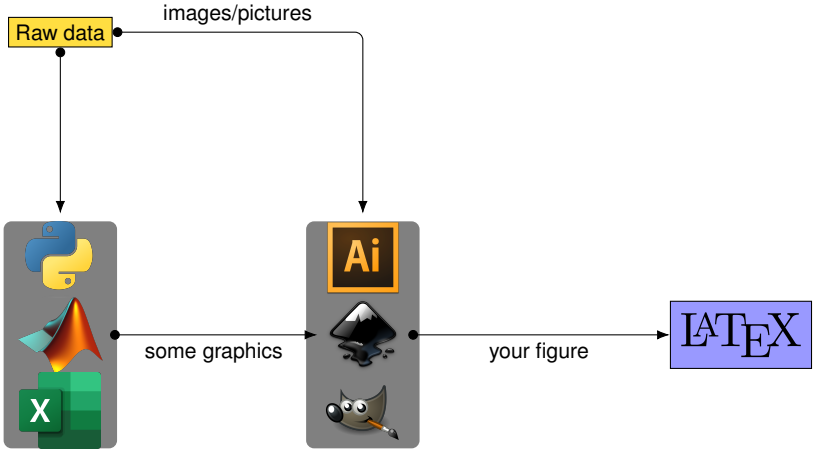
- All figures are different, it is hard to establish a consistent workflow
- Scientific figures are subject to multiple changes before publication
- They often rely on multiple processes over different software

For all those reasons, establishing a robust and efficient workflow for scientific figures might be **very** complicated!

Example of common workflow

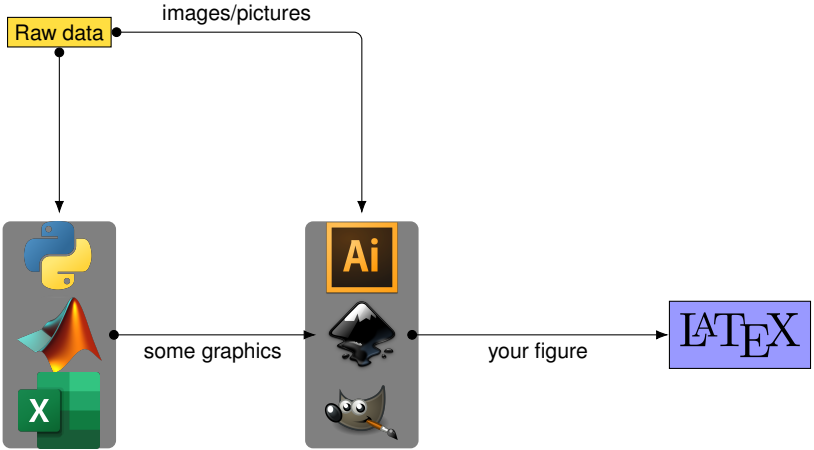


Example of common workflow



J-2 to submission your supervisor/colleague tells you:
"This plot should be logscale, with dashed lines and the figure should be bigger."

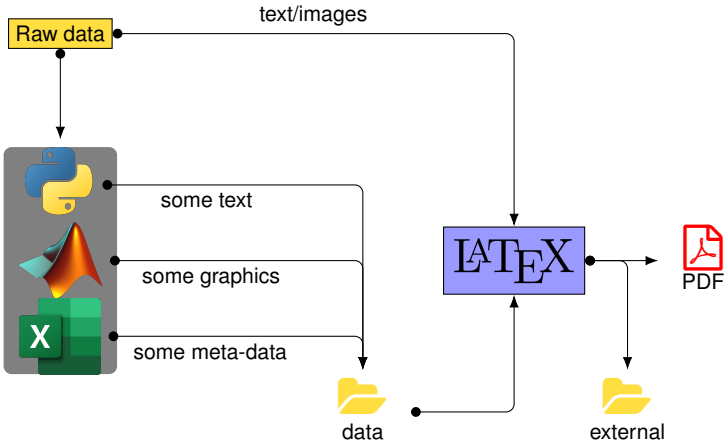
Example of common workflow



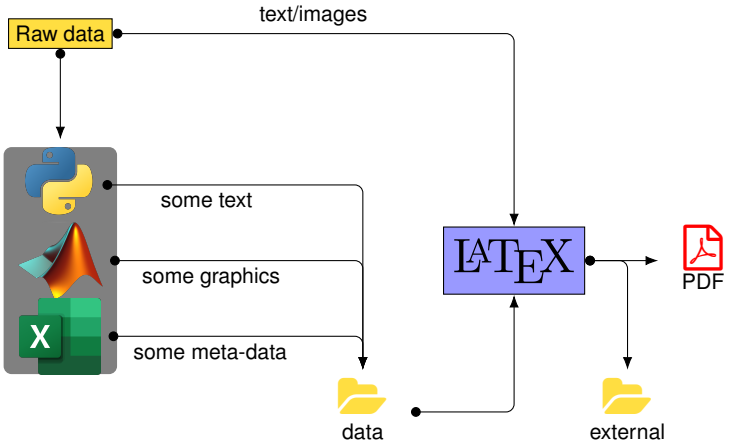
Some of the bottlenecks:

Software versions & licenses, mismatch with LaTeXfonts/colors/styles, high-sensitivity to small changes, nested operations, ...

Proposed workflow

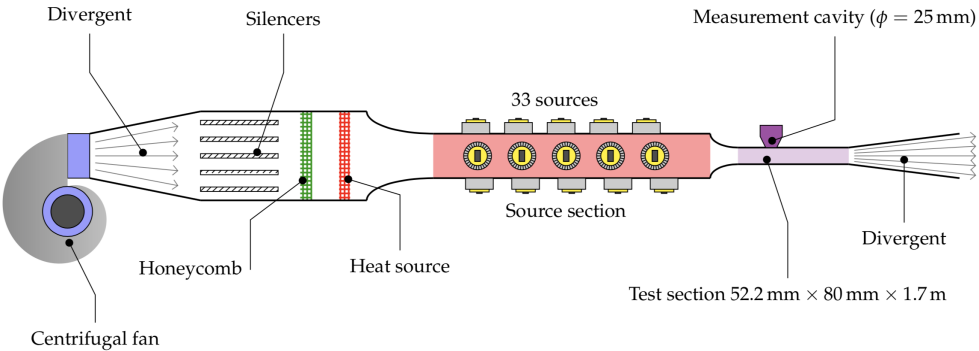


Proposed workflow

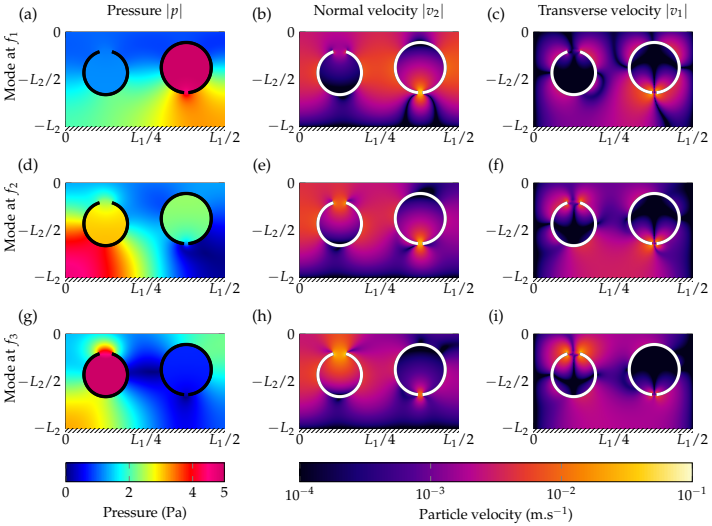


Solution: LaTeXis free, across all operating systems, fully implemented online through Overleaf, no third-party software, low-sensibility to changes, vectorized outputs

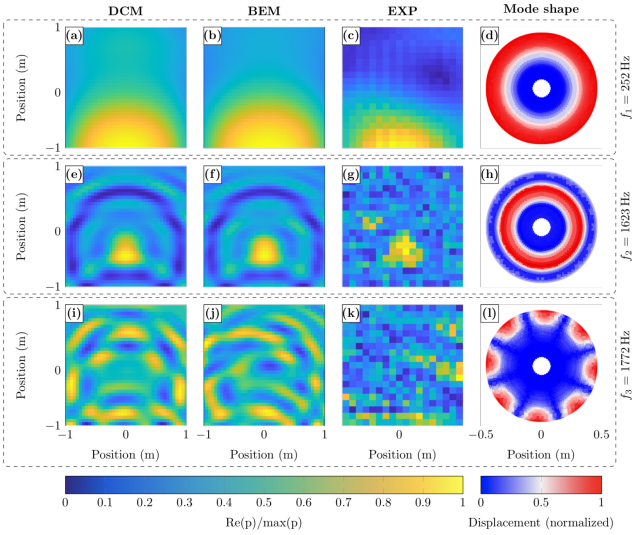
Some examples



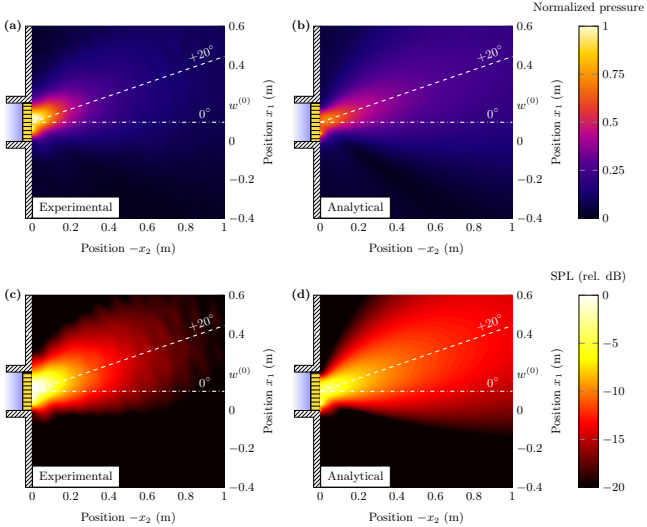
Some examples



Some examples

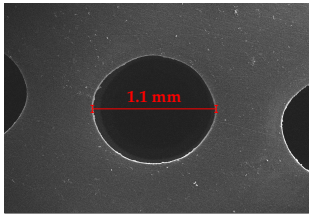


Some examples

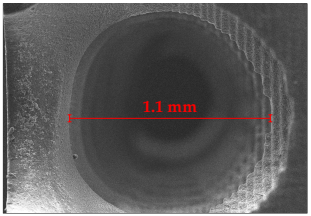


Some examples

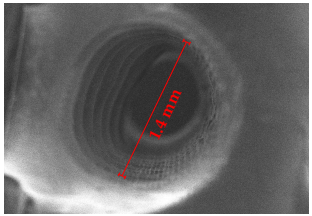
(a)



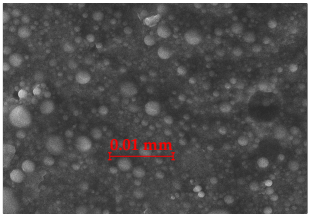
(b)



(c)



(d)



Should you use it?

Advantages:

- Entirely free and accessible online!
- Data and figure generation are separated, limits intermediate steps
- The output illustrations are vector graphics
- Everything is done in a single software: your IDE/editor

Should you use it?

Advantages:

- Entirely free and accessible online!
- Data and figure generation are separated, limits intermediate steps
- The output illustrations are vector graphics
- Everything is done in a single software: your IDE/editor

Drawbacks:

- No mouse usage since it's only code
- Lots of time debugging (hint: it's always your fault)
- Not really meant for 3D – use Blender instead

Packages

Few packages are required ...

TeX-editor

```
\usepackage{tikz}
\usepackage{pgfplots}
```

Packages

Few packages are required ...

TeX-editor

```
\usepackage{tikz}
\usepackage{pgfplots}
```

... but you may need more depending on how sophisticated your illustrations are :)

TeX-editor

```
%-----------------------------%
% TIKZ PACKAGES
%-----------------------------%
\usepackage{tikz}
\usetikzlibrary{patterns}
\usetikzlibrary{arrows.meta}
\usetikzlibrary{calc,shadows,matrix}
\usetikzlibrary{decorations.pathreplacing}
\usetikzlibrary{decorations.pathmorphing}
\usetikzlibrary{automata,positioning}
```

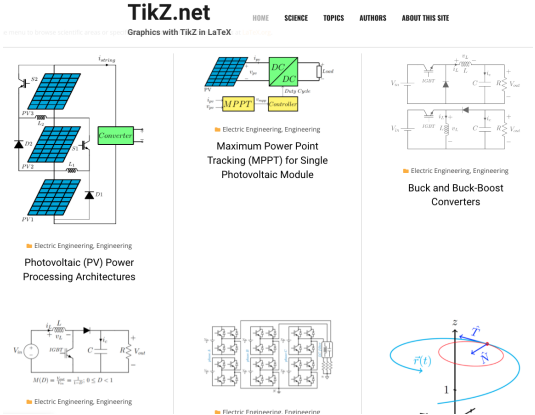
TeX-editor

```
%-----------------------------%
% PGF PACKAGES
%-----------------------------%
\usepackage{pgfplots}
%\usepgfplotslibrary{external}
%\tikzexternalize[optimize=false]
%\tikzsetexternalprefix{external/}
\usepgfplotslibrary{patchplots}
\usepgfplotslibrary{fillbetween}
```

Need some documentation?

Two main packages: **TikZ & PGFPlots** – They come with lots of subpackages

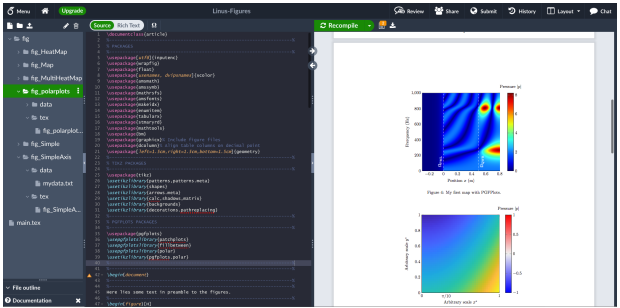
- `ctan.org`
- `tikz.net`
- TikZ pour l'impatient (in French)
- stackexchange & forums



Overleaf - Free Online IDE

Overleaf

Connect to `overleaf.com` and log in



- Free online editor and compiler
- No configuration needed
- Provides many templates
- Possibility to collaborate

Starting up

Steps to follow:

- Connect to overleaf.com
- Log in with your email address
- Create a new blank project
- Connect to `https://www.lib4ri.ch/searching-managing-scientific-information`
- Search for Module 5+
- Download the `.zip` file on your local machine
- Extract the `.zip` file
- Drag & drop the file `main.tex` in the project explorer on Overleaf

Starting a TikZ figure

TikZ comes with a set of basic and intuitive instructions. First, let's enter the `figure` environment and create a `tikzpicture`:

TeX-editor

```
\begin{figure}
\centering                          %Centers the figure
\begin{tikzpicture}
...
...                                 %Your instructions here
\end{tikzpicture}
\caption{My caption here.}          %Figure Caption
\label{fig:First}                   %Label of the figure
\end{figure}
```

Now we can start to execute some instructions within the `tikzpicture` environment.

Starting a TikZ figure

One of the simplest command to execute is `draw`, such as:

TeX-editor

```
\begin{tikzpicture}
\draw [some style] (x1,y1) <some shape> (x2,y2);
\end{tikzpicture}
```

Starting a TikZ figure

One of the simplest command to execute is `draw`, such as:

TeX-editor

```
\begin{tikzpicture}
\draw [black] (0,0) rectangle (1,2);    %black rectangle of size 1x2
\end{tikzpicture}
```

Starting a TikZ figure

One of the simplest command to execute is `draw`, such as:

TeX-editor

```
\begin{tikzpicture}
\draw [black] (0,0) rectangle (1,2);    %black rectangle of size 1x2
\end{tikzpicture}
```

which outputs the following shape:

Starting a TikZ figure

One of the simplest command to execute is `draw`, such as:

TeX-editor

```
\begin{tikzpicture}
\draw [black] (0,0) rectangle (1,2);    %black rectangle of size 1x2
\end{tikzpicture}
```

which outputs the following shape:

Now let's play with the styles of our rectangle!

Starting a TikZ figure

Changing the styles of the rectangle with the `[style]` arguments:

TeX-editor

```
\begin{tikzpicture}
%black rectangle of size 1x2
\draw [black] (0,0) rectangle (1,2);
\end{tikzpicture}
```

Starting a TikZ figure

Changing the styles of the rectangle with the `[style]` arguments:

TeX-editor

```
\begin{tikzpicture}
%black, dashed
\draw [black,dashed] (0,0) rectangle (1,2);
\end{tikzpicture}
```

Starting a TikZ figure

Changing the styles of the rectangle with the `[style]` arguments:

TeX-editor

```
\begin{tikzpicture}
%black, dashed, orange fill
\draw [black,dashed,fill=orange] (0,0) rectangle (1,2);
\end{tikzpicture}
```

Starting a TikZ figure

Changing the styles of the rectangle with the `[style]` arguments:

TeX-editor

```
\begin{tikzpicture}
%black, dashed, orange fill, thick edge
\draw [black,dashed,fill=orange,line width=2pt] (0,0) rectangle (1,2);
\end{tikzpicture}
```

Starting a TikZ figure

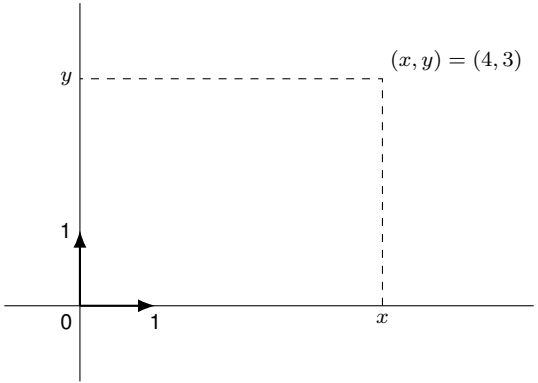
Changing the styles of the rectangle with the `[style]` arguments:

TeX-editor

```
\begin{tikzpicture}
%black, dashed, orange fill, rounded corners
\draw [black,dashed,fill=orange,rounded corners] (0,0) rectangle (1,2);
\end{tikzpicture}
```

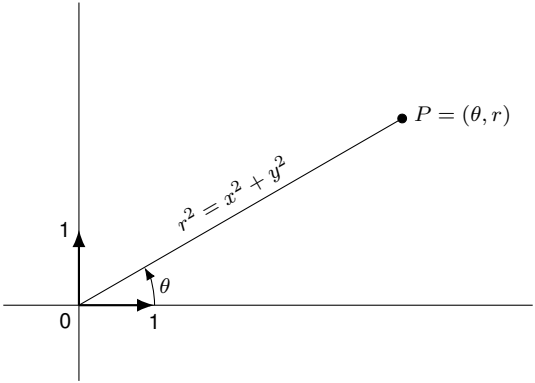
Positions on the figure

But, where the origin is located in the generated figure? In fact, it has no predefined position. The figure occupies only the minimum space so that all the elements drawn explicitly are visible. The position of the origin in relation to the figure therefore depends on the points drawn.



Positions on the figure

Until now, we have worked with Cartesian coordinates. Coordinates are given within parentheses such as (3,4). TikZ also supports polar coordinates!



Pythagoras gives $(\theta, r) \equiv (r\cos(\theta), r\sin(\theta))$, don't hesitate to use both polar and Cartesian systems!

Coordinates

The coordinates can be specified explicitly using the command `\coordinate`.

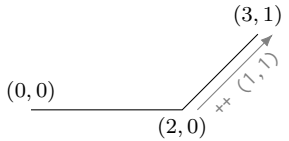
TeX-editor

```
\begin{tikzpicture}
\coordinate (0) at (0,0);        %coordinate (0)
\draw (0) -- (2,0);              %horizontal segment
\end{tikzpicture}
```

In addition, relative coordinates can become very useful

TeX-editor

```
\begin{tikzpicture}
\coordinate (0) at (0,0);                %coordinate (0)
\draw  (0) -- (2,0) -- ++ (1,1);         %multiple segments
\end{tikzpicture}
```



Circles and arcs

TikZ comes with additional shapes, namely the `circle` and the `arc`

TeX-editor

```
\begin{tikzpicture}
\draw (0,0) circle (0.5);        %draw circle with r=0.5 at (0,0)
\end{tikzpicture}
```

TeX-editor

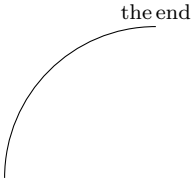
```
\begin{tikzpicture}
\draw (0,0) -- (1,0) arc (180:0:0.5) -- ++ (1,0);        %segment, then arc, then segment
\end{tikzpicture}
```

Writing text and labels

In your illustrations, you can add text and labels using `nodes`.

TeX-editor

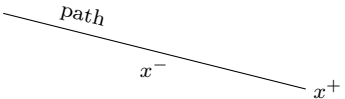
```
\begin{tikzpicture}
\draw (0,0) arc(180:90:2) node[pos=1,above]{the end};
\end{tikzpicture}
```



You can specify the position of the node relative to the current object drawn.

TeX-editor

```
\begin{tikzpicture}
\draw (0,0) -- (4,-1)
node[pos=1,right]{$x^+$}
node[midway,below]{$x^-$}
node[near start,above,sloped]{path};
\end{tikzpicture}
```

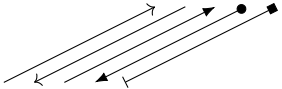


Arrows and curves

The line segments can be decorated, to make arrows for example This is specified as arguments of the `\draw` command.
This useful to place labels or indicate directions within your illustrations.

TeX-editor

```
\begin{tikzpicture}
\draw [->] (0.4,0) -- ++ (2,1);
\draw [<-] (0.8,0) -- ++ (2,1);
\draw [-Latex] (1.2,0) -- ++ (2,1);
\draw [Latex-Circle] (1.6,0) -- ++ (2,1);
\draw [Bar-Square] (2,0) -- ++ (2,1);
\end{tikzpicture}
```



More arrow tips and styles are available here: tikz.dev/tikz-arrows
More decorations and styles can be found here: tikz.dev/library-decorations

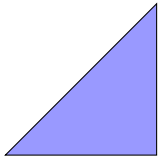
Now you should have enough tools to start working on your very first figure! Try to redo the ones on slides 15-16.

Closed curves and fills

Using the instruction `cycle` points towards the start of the current path. With the command `\fill` you can directly specify the filling colour.

TeX-editor

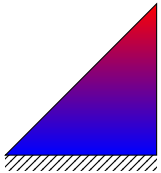
```
\begin{tikzpicture}
\fill [blue!40,draw=black] (0,0) -- (2,0) -- (2,2) -- cycle;
\end{tikzpicture}
```



Similarly, the command `\shade` lets you apply colour gradients to your shapes!

TeX-editor

```
\begin{tikzpicture}
\shade [top color=red,bottom color=blue,draw=black]
(0,0) -- (2,0) -- (2,2) -- cycle;
\fill [pattern=north east lines] (0,0) rectangle (2,-.2);
\end{tikzpicture}
```

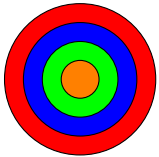


Colouring

The colors in your sketch can be changed as arguments of the `\draw` command. The basic colors are intuitive red, blue, green, orange, black ...

TeX-editor

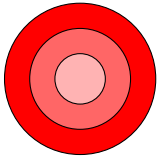
```
\begin{tikzpicture}
\draw [fill=red] (0,0) circle (1);
\draw [fill=blue] (0,0) circle (.75);
\draw [fill=green] (0,0) circle (.5);
\draw [fill=orange] (0,0) circle (.25);
\end{tikzpicture}
```



You can also blend-in or fade colors as such:

TeX-editor

```
\begin{tikzpicture}
\draw [fill=red] (0,0) circle (1);
\draw [fill=red!60] (0,0) circle (2/3);
\draw [fill=red!30] (0,0) circle (1/3);
\end{tikzpicture}
```



More colours here: en.wikibooks.org/wiki/LaTeX/Colors

Variables

In addition, LaTeX can also handle variables with the command `\def` as shown below:

TeX-editor

```
\begin{tikzpicture}
\def\Lx{1};                          %define variable Lx
\def\Ly{2};                          %define variable Ly
\coordinate (O) at (0,0);            %coordinate (O)
\coordinate (A) at (\Lx,\Ly);        %coordinate (A)
\draw [black,dashed] (O) rectangle (A); %draw rectangle
\end{tikzpicture}
```

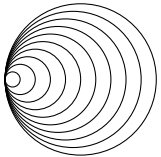
Defining variables becomes extremely useful to build your illustration, especially when combined with relative coordinates.

Loops

TikZ provides the command `\foreach` to make loops.

TeX-editor

```
\begin{tikzpicture}
\foreach \k in {1,...,10}
{\draw (\k/10,0) circle (\k/10);}
\end{tikzpicture}
```



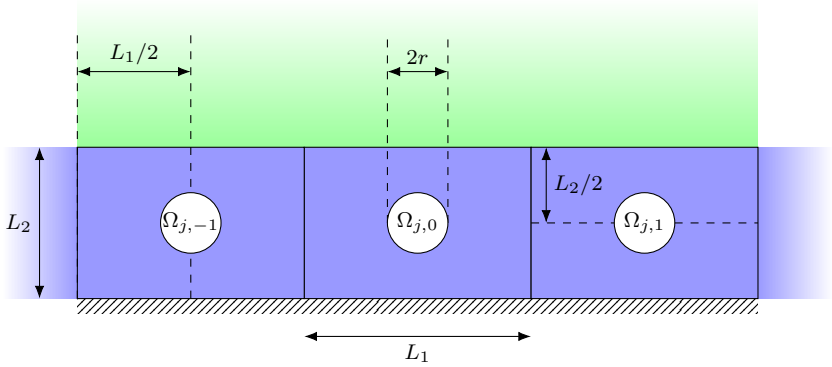
TeX-editor

```
\begin{tikzpicture}
\foreach \k in {1,...,4}
{
\foreach \m in {1,...,3}
{
\draw (\k,\m) rectangle ++ (.75,.75) node[midway]{\k,\m};
}
}
\end{tikzpicture}
```

1,3	2,3	3,3	4,3
1,2	2,2	3,2	4,2
1,1	2,1	3,1	4,1

Exercise

Now it is your turn to work on a second exercise. It has `rectangles`, `circles`, `arrows`, and `shades`! You can use variables and loops (or not). The colors are `blue!40` and `green!40`.



Plotting data

Creating plots is made easy with the `axis` environment:

TeX-editor

```
\begin{tikzpicture}
\begin{axis}[some style]
\addplot [some style] <type> {data};            %Your data goes here
\end{axis}
\end{tikzpicture}
```

Plotting data

Creating plots is made easy with the `axis` environment:

TeX-editor

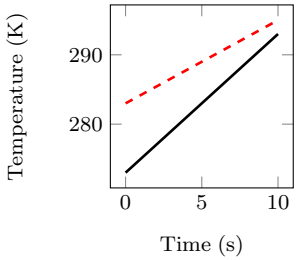
```
\begin{tikzpicture}
\begin{axis}[height=4cm, width=4cm,xlabel=Time (s),ylabel=Temperature (K)]
\addplot [solid,black,line width=1pt] coordinates {(0,273) (10,293)};   %Temperature data1
\addplot [dashed,red,line width=1pt] coordinates {(0,283) (10,295)};    %Temperature data2
\end{axis}
\end{tikzpicture}
```

Plotting data

Creating plots is made easy with the `axis` environment:

TeX-editor

```
\begin{tikzpicture}
\begin{axis}[height=4cm, width=4cm,xlabel=Time (s),ylabel=Temperature (K)]
\addplot [solid,black,line width=1pt] coordinates {(0,273) (10,293)};   %Temperature data1
\addplot [dashed,red,line width=1pt] coordinates {(0,283) (10,295)};    %Temperature data2
\end{axis}
\end{tikzpicture}
```



Making subplots

PGFPlots also handles multiple plots, which can be explicitly placed relative to each other.

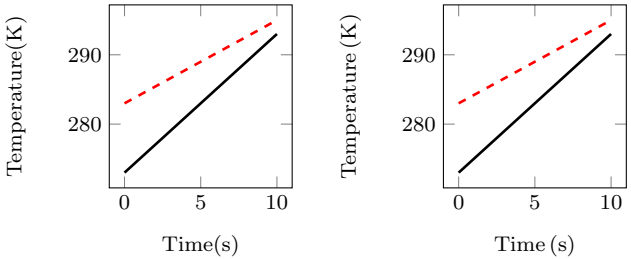
TeX-editor

```
%-----------------------------------------------%
\begin{tikzpicture}
%-----------------------------------------------%
\begin{axis}[name=plot1,height=4cm, width=4cm,
xlabel=Time (s),ylabel=Temperature (K)]
\addplot [solid,black,line width=1pt] coordinates {(0,273) (10,293)};   %Temperature data1
\addplot [dashed,red,line width=1pt] coordinates {(0,283) (10,295)};    %Temperature data2
\end{axis}

\begin{axis}[name=plot2,height=4cm, width=4cm,          %Plot name and dimension
at={($(plot1.east) + (2cm,0)$)},anchor=west,            %Plot position and anchor
xlabel=Time (s),ylabel=Temperature (K)]                 %Plot labels
\addplot [solid,black,line width=1pt] coordinates {(0,273) (10,293)};   %Temperature data1
\addplot [dashed,red,line width=1pt] coordinates {(0,283) (10,295)};    %Temperature data2
\end{axis}
%-----------------------------------------------%
\end{tikzpicture}
%-----------------------------------------------%
```

Making subplots

PGFPlots also handles multiple plots, which can be explicitly placed relative to each other.



The position of the subplots is controlled with the arguments `at` and `anchor`.
With `at` you specify **where** the plot `anchor` should be: in this case, the `west` anchor of `plot2` is 2cm on the left of `plot1.east`.

Controlling axis and dimension

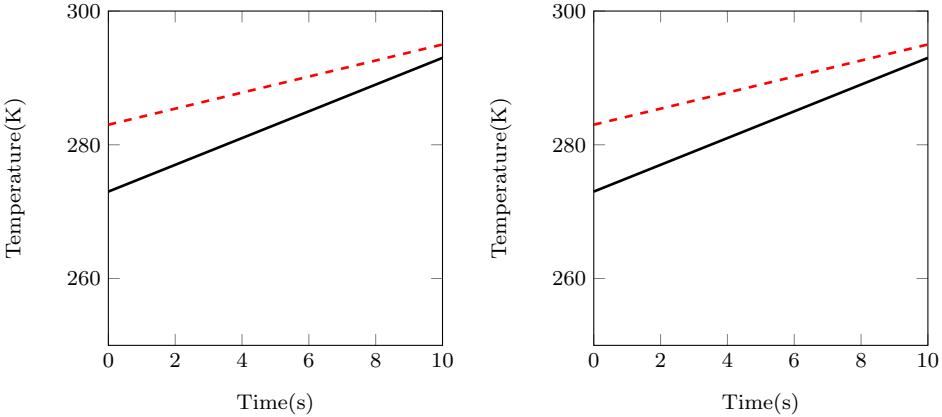
TeX-editor

```
\begin{tikzpicture}
%-----------------------------------------------%
\def\height{6cm};
\def\width{6cm};
\def\spx{2cm};

\begin{axis}[name=plot1,height=\height, width=\width,
xmin=0,xmax=10,ymin=250,ymax=300,
xlabel=Time (s),ylabel=Temperature (K)]
\addplot [solid,black,line width=1pt] coordinates {(0,273) (10,293)};   %Temperature data1
\addplot [dashed,red,line width=1pt] coordinates {(0,283) (10,295)};    %Temperature data2
\end{axis}

\begin{axis}[name=plot2,height=\height, width=\width,           %Plot name and dimension
xmin=0,xmax=10,ymin=250,ymax=300,
at={($(plot1.east) + (\spx,0)$)},anchor=west,           %Plot position and anchor
xlabel=Time (s),ylabel=Temperature (K)]                 %Plot labels
\addplot [solid,black,line width=1pt] coordinates {(0,273) (10,293)};   %Temperature data1
\addplot [dashed,red,line width=1pt] coordinates {(0,283) (10,295)};    %Temperature data2
\end{axis}
%-----------------------------------------------%
\end{tikzpicture}
```

Controlling axis and dimension



Defining variables for `height`, `width` and `spacing` is very convenient to place your plots consistently.

Adding legends

The legends are easily controlled with the command `legend`:

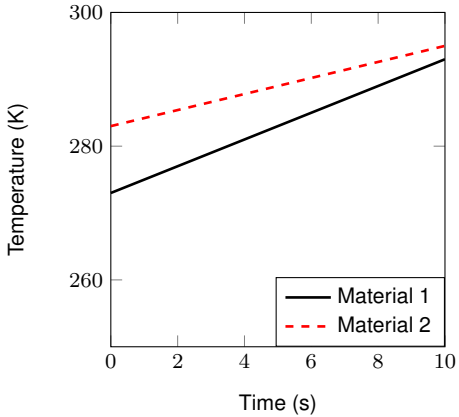
TeX-editor

```
\begin{tikzpicture}
%-------------------------------------------%
\def\height{6cm};
\def\width{6cm};
\def\spx{2cm};

\begin{axis}[name=plot1,height=\height, width=\width,
xmin=0,xmax=10,
ymin=250,ymax=300,
legend entries={Material 1, Material 2},
legend style={at=(1,0),anchor=south east},
xlabel=Time (s),ylabel=Temperature (K)]
\addplot [solid,black,line width=1pt] coordinates {(0,273) (10,293)};   %Temperature data1
\addplot [dashed,red,line width=1pt] coordinates {(0,283) (10,295)};    %Temperature data2
\end{axis}
%-------------------------------------------%
\end{tikzpicture}
```

Adding legends

The legends are easily controlled with the command `legend`:



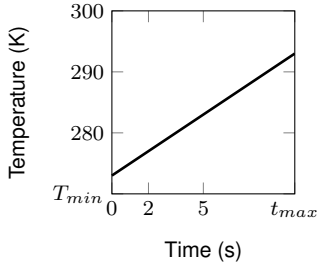
The `legend entries` must be comma separated, and follow the order in which your plots are drawn on the `axis` environment.

Ticks and labels

The `ticks` and `ticklabels` can be manually specified

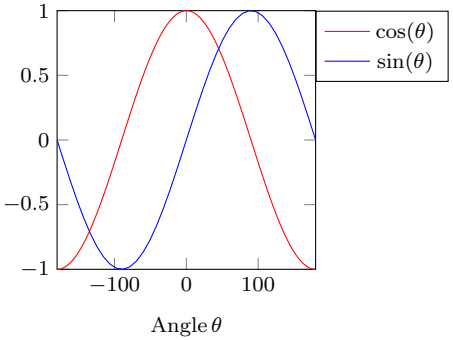
TeX-editor

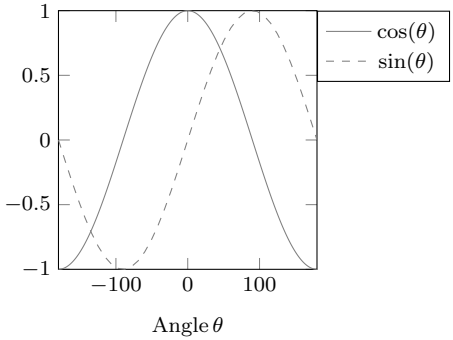
```
\begin{tikzpicture}
\begin{axis}[height=4cm, width=4cm,xlabel=Time (s),ylabel=Temperature (K),
xtick={0,2,5,10},xticklabels={$0$,$2$,$5$,$t_{max}$},
ytick={270,280,290,300},yticklabels={$T_{min}$,$280$,$290$,$300$},
xmin=0,xmax=10,ymin=270,ymax=300]
\addplot [solid,black,line width=1pt] coordinates {(0,273) (10,293)};   %Temperature data1
\end{axis}
\end{tikzpicture}
```



Coloring and line style

It is important to cycle between **different line styles and colors** and to have a meaningful legend. Also, try to decimate the information you are plotting, i.e., you don't need 20'000 data points for a 5 cm plot.

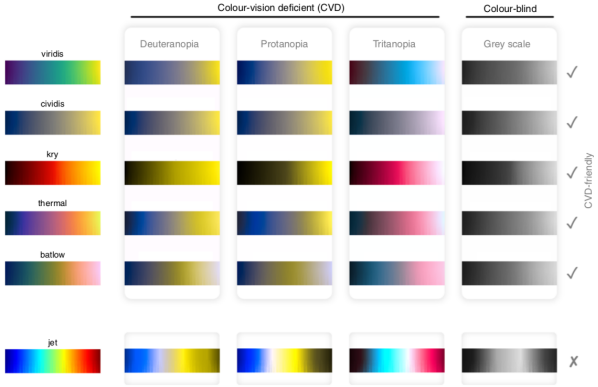




Most of the scientific literature is printed on black& white paper, you should adapt your colouring. The common line styles are `solid`, `dashed`, `dotted`, `dashdotted`, `...` you can also make use of markers.

Coloring and line style

This is also important when dealing with colormaps and more advanced data representations.
Fact: Colour vision deficiency affects 1 in 12 men and 1 in 200 women ... so keep an eye on the colors you use to represent your datasets!



Read paper here: Crameri, *et. al.*, 2020

A word about captioning

- Give an explicit and rich description of your illustration.
- Use labels and directions if needed
- Mention the different subfigures with nodes **(a)**, **(b)** ...

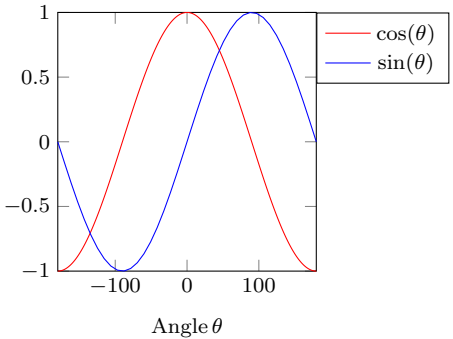


Figure 1: Analytical results

A word about captioning

- Give an explicit and rich description of your illustration.
- Use labels and directions if needed
- Mention the different subfigures with nodes **(a)**, **(b)** ...

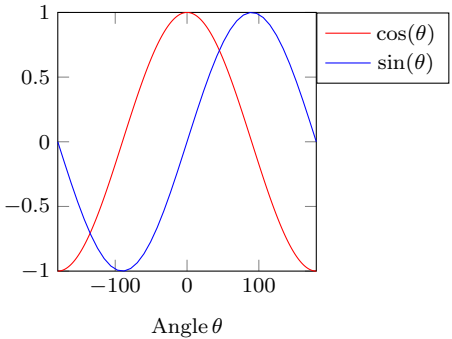


Figure 1: Analytical results, amplitude of the cosine function (red) and of the sine function (blue) with respect to the angle θ.

Importing data

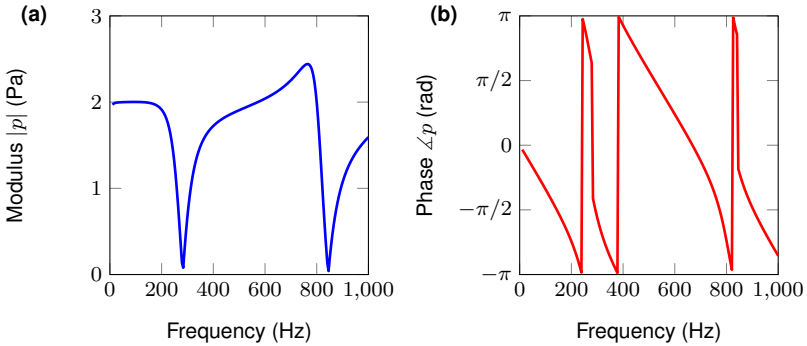
Now that we can make nice plots in LaTeX, how to import real data? Instead of plotting `coordinates`, we can point to a `file` that contains data. This is performed using the following command:

TeX-editor

```
%-----------------------------------------------%
\begin{tikzpicture}
%-----------------------------------------------%
\begin{axis}[axis arguments]                          %labels, ticks, ...
\addplot [plot style] file {/mypath/myfile.txt};      %plot data from file
\end{axis}
%-----------------------------------------------%
\end{tikzpicture}
%-----------------------------------------------%
```

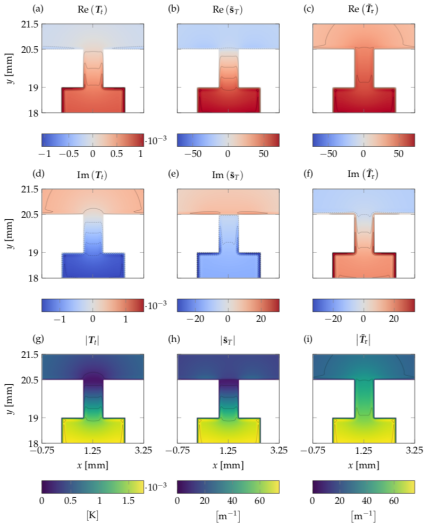
Importing data

You can exercise on this figure! The data is provided in the `.zip` file downloaded earlier.



Plotting maps

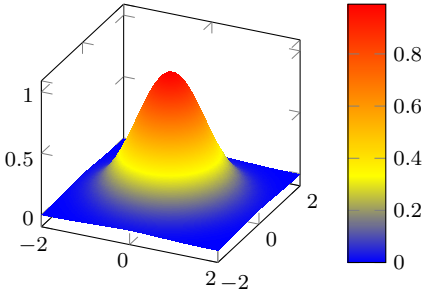
- Let `PGFPlots` do the job
  - Pros: No pre-processing needed, colormap-independent
  - Cons: Limited memory usage, longer compilation time
- Export high-DPI `png`
  - Pros: No memory issues, adjustable resolution
  - Cons: Raster image (although invisible), colormap-dependent, shorter compilation time



Plotting maps – using \addplot3

TeX-editor

```
\begin{tikzpicture}
\begin{axis}[
height=5cm,width=5cm,
colorbar right,shader=interp]
\addplot3[surf,domain=-2:2,samples=30] {exp(-x^2-y^2)};
\end{axis}
 \end{tikzpicture}
```

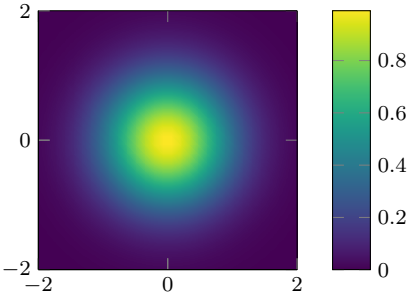


Plotting 3D maps is not always a good idea, as the projection on a 2D canvas hides part of the information!

Plotting maps – using \addplot3

TeX-editor

```
\begin{tikzpicture}
\begin{axis}[
height=5cm,width=5cm,
view={0}{90},
colormap/viridis,
colorbar right,shader=interp]
\addplot3[surf,domain=-2:2,samples=30] {exp(-x^2-y^2)};
\end{axis}
 \end{tikzpicture}
```



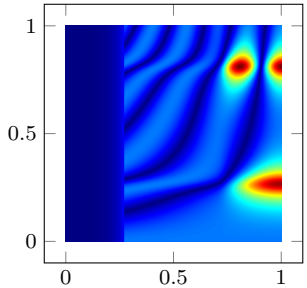
Then setting the camera to the top with gives more visibility over the whole data. Also, remember to include a `colorbar` and specify the `colormap`!

Plotting maps – image import

As memory can become an issue, a common practice is to import the maps as png directly!

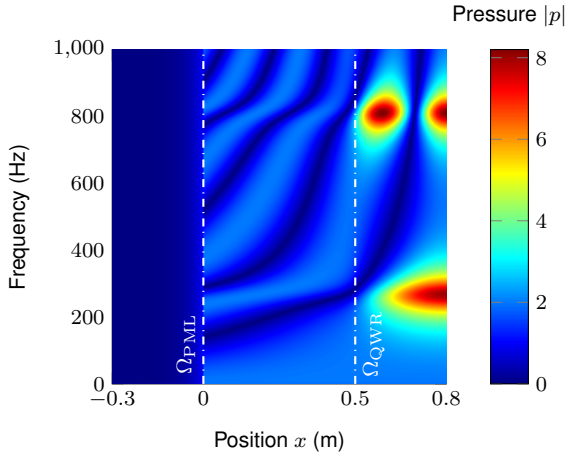
TeX-editor

```
\begin{tikzpicture}
\begin{axis}[some arguments]
\addplot [forget plot] graphics [bounds] {path/file.png};
\end{axis}
\end{tikzpicture}
```

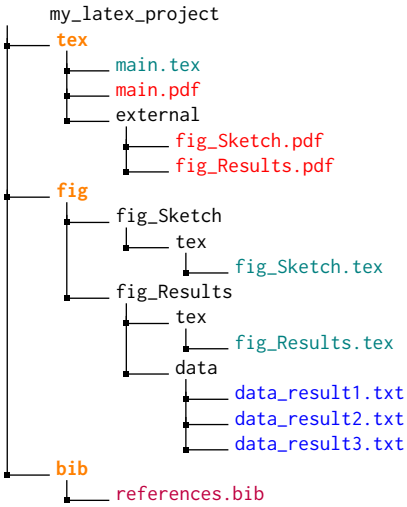


Plotting maps – image import

Exercise: Try to redo the following figure!

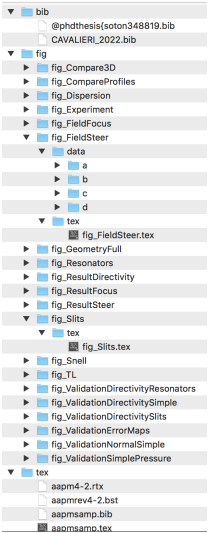


Organising your files – a rigorous structure



Organising your files – a rigorous structure

```
my_latex_project
├── tex
│   ├── main.tex
│   ├── main.pdf
│   └── external
│       ├── fig_Sketch.pdf
│       └── fig_Results.pdf
├── fig
│   ├── fig_Sketch
│   │   └── tex
│   │       └── fig_Sketch.tex
│   └── fig_Results
│       ├── tex
│       │   └── fig_Results.tex
│       └── data
│           ├── data_result1.txt
│           ├── data_result2.txt
│           └── data_result3.txt
└── bib
    └── references.bib
```



TikZ externalization

As you add more and more figures to the compiler queue, the process can become extensively long. Externalization allows to pre-compute and store figures that have been untouched since previous compilation. In the preamble to your document, use the following instructions:

TeX-editor

```
%-----------------------------------------------%
\usepackage{tikzexternalize}                          %import package
\tikzsetexternalprefix{external/}                     %define /external folder
%-----------------------------------------------%
\newcommand{\external}[1]{\tikzsetfigurename{#1}}     %\external{fig_name} command
\newcommand{\externalremake}{\tikzset{external/remake next}}    %force re-compile (optional)
%-----------------------------------------------%
```

Set-up externalization in your LaTeX project

On overleaf: Create the `/external` folder with a dummy file `dummy.txt`. The output figures are stored internally and are accessible within "`Logs and output files`".

On your local machine: Have the compiler set up with the option `-shell-escape`. Create the `/external` folder.

TikZ externalization

Tips for organising your project

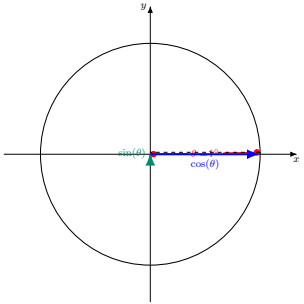
- Use meaningful names for your figures
- Unify the name for folders, files, labels ...
- Define a shortcut in your IDE to write these instructions

Format each `figure` environment as follows:

TeX-editor

```
%-----------------------------------------------%
\begin{figure}[H]
\centering                                      %center the figure
\external{fig_MyTest}                           %name for externalization
%\externalremake                                %uncomment to force compilation
\input{../fig/fig_MyTest/tex/fig_MyTest}        %input the figure's .tex file
\caption{This is a test caption.}               %caption to the figure
\label{fig:MyTest}                              %label of the current figure
\end{figure}
%-----------------------------------------------%
```

Animation within PDF!



Animation within PDF!

Making use of the `animate` package as such:

TeX-editor

```
\usepackage{animate}                          %import animate package
%---------------------------------------------%
\begin{figure}                                %open figure environment
\setcounter{m}{1}                             %initial counter to m=1
%---------------------------------------------%
\begin{animateinline}[style]{fps}             %open animateinline environment
\whiledo{\them < Nframes}{                    %while m < Nframes
\begin{tikzpicture}[scale=1]                  %open tikzpicture envirnoment
% TikZ instructions here                      %draw stuff
\end{tikzpicture}
\stepcounter{m}                               %m = m+1
\ifthenelse{\them < Nframes}
{\newframe}{\end{animateinline}\relax}}       %if m < Nframes: continue // else: stop
%---------------------------------------------%
\end{figure}
%---------------------------------------------%
```

Quote

You are not simply "writing" a document – you are crafting a work of art that combines both the precision and creativity of your logical thinking, as well as the elegance of a beautifully typeset writing.

Quote

You are not simply "writing" a document – you are crafting a work of art that combines both the precision and creativity of your logical thinking, as well as the elegance of a beautifully typeset writing. The amount of attention and care you put into the presentation is indicative of the amount of thought you put into the content.

Quote

You are not simply "writing" a document – you are crafting a work of art that combines both the precision and creativity of your logical thinking, as well as the elegance of a beautifully typeset writing. The amount of attention and care you put into the presentation is indicative of the amount of thought you put into the content. Therefore, having good style is not only delightful and aesthetically pleasing to read, but it also serves to establish your ethos and character.

Quote

You are not simply "writing" a document – you are crafting a work of art that combines both the precision and creativity of your logical thinking, as well as the elegance of a beautifully typeset writing. The amount of attention and care you put into the presentation is indicative of the amount of thought you put into the content. Therefore, having good style is not only delightful and aesthetically pleasing to read, but it also serves to establish your ethos and character. One can tell that someone puts a lot of effort into their work and takes great pride in them when they pay attention even to the smallest of details. – Fan Pu Zeng

Quote

You are not simply "writing" a document – you are crafting a work of art that combines both the precision and creativity of your logical thinking, as well as the elegance of a beautifully typeset writing. The amount of attention and care you put into the presentation is indicative of the amount of thought you put into the content. Therefore, having good style is not only delightful and aesthetically pleasing to read, but it also serves to establish your ethos and character. One can tell that someone puts a lot of effort into their work and takes great pride in them when they pay attention even to the smallest of details. – Fan Pu Zeng

[From French] Methods are habits of the mind and spares of memory. – Antoine de Rivarol