NTNU
Kunnskap for en bedre verden

Department of Engineering Cybernetics

Paper Title

Authors:
Firstname Lastname

February 12, 2021

Abstract

In the abstract, give the reader an overview of the paper. Briefly state the purpose of this project, how it was conducted, and summarize your key findings.

Amount: 0.5 page

Table of Contents

List of Figures

List of Tables

1 Introduction

2 Theory

3 Method

4 Result

5 Conclusion

Bibliography

[1] U. Ghia, K. N. Ghia, and C. T. Shin. High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method. *Journal of Computational Physics*, 48:387–411, 1982.

[2] Carl Sagan. *Brocas brain: reflections on the romance of science.* Presidio Press, 1993.

[3] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1995–2003. JMLR.org, 2016.

Appendix

These appendices contain examples of common use cases that hopefully might prove useful as reference material, especially for those new to LaTeX. Assuming you're working in the editor of overleaf.com, the markup for all the examples below can be found by clicking on `example_appendix.tex` under the `Appendices` folder in the left hand project overview.

A References

When writing, you often have to refer back to something you said earlier, forward to something you will get back to later, to figures and tables, or to external sources.

A.1 Within the Document

Sections, figures, tables, and pretty much anything in a latex document can be labeled and cross-referenced. For instance, this (subsub)section is labeled with `\label{app:within-the-document}`, allowing it to be referenced throughout the document. The advantage of labeling and referencing is that it allows the correct element code (A1 at the time of writing) to be generated at compile time, which in turn reduces the chance that old references become outdated as the project evolves over time.

After an element is labeled, it can be referenced in several ways:

- Reference element code: `\ref{app:within-the-document}` → A.1
- Reference element page `\pageref{app:within-the-document}` → 3
- Autoformat reference[1]: `\autoref{app:within-the-document}` → section A.1

A.2 Citations

Referencing to external work, in the form of citations, requires two steps. First, descriptions of the relevant work must be made available. In this project, the content of "references.bib" will be taken into account when compiling the project. This file can be populated manually, which is fine for smaller projects with only a handful of citations. Alternatively, if you're using a reference manager such as Zotero or Mendeley to organize your collection of pdfs, they typically come with a function to export a .bib file that can be uploaded to the project. However, the process can be further streamlined by connecting your reference manager directly to overleaf (see `https://www.overleaf.com/blog/639-tip-of-the-week-overleaf-and-reference-managers`).

Once the bibliography is made available, it can be cited:

- `\cite{ghia_et_al}` → [1]
- `\citep{ghia_et_al}` → [1]
- `\citep{ghia_et_al, sagan_1993}` → [1, 2]

Cited bibliography will automatically show up in the bibliography section of this document. The exact appearance depends on the configured citation style, which at the time of writing is set to `plainnat` (see `https://www.overleaf.com/learn/latex/Natbib_bibliography_styles` for alternatives).

[1] The exact format of autoref is configurable. See "setup.sty".

B Figures

Beautiful figures will make your report pop. The subsequent examples cover some of the most common ways to insert them into the document.

B.1 Simple Figures

Figures are declared with the `\being{figure}...\end{figure}` command. You can put different forms of graphics inside it, but the most common use case involves loading graphics from an external image file. Similarly to sections, images can be labeled and referenced throughout the document (see Figure 1).

Note: If you did not produce the figure content yourself, remember to credit the original author.

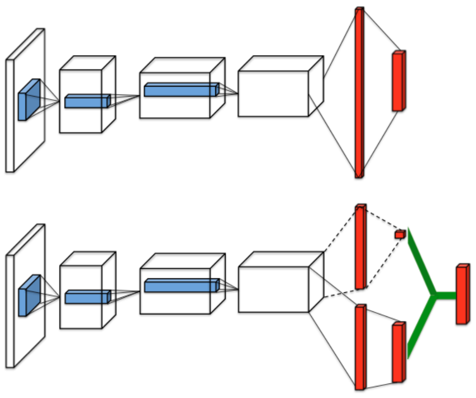
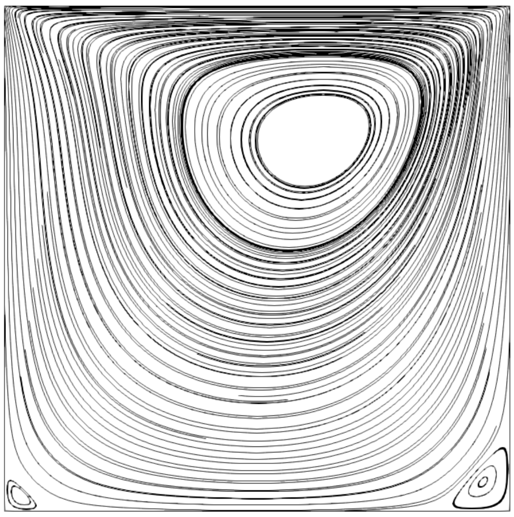


Figure 1: Example figure

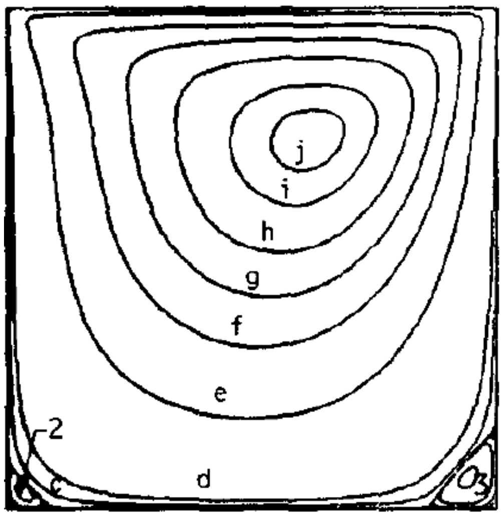
Source: [3]

B.2 Sub-figures

It is also possible to compose a single figure from several graphical elements. Figure 2 gives an example of how this can be done with the use of subfigures.



Re = 100 - Testing method.



Re = 100 - Ghia et al. [1].

Figure 2: Example sub-figures

B.3 Generated Diagrams

There are also several packages for generating figures and diagrams. Figure 3 was generated with the `tikz` package, which allows drawing arbitrary vector graphics with LaTeX commands. While elegant, learning it might take some time, so an easier solution is to use conventional drawing tools with graphical user interfaces (e.g. `draw.io`) and display the resulting figures as conventional images.

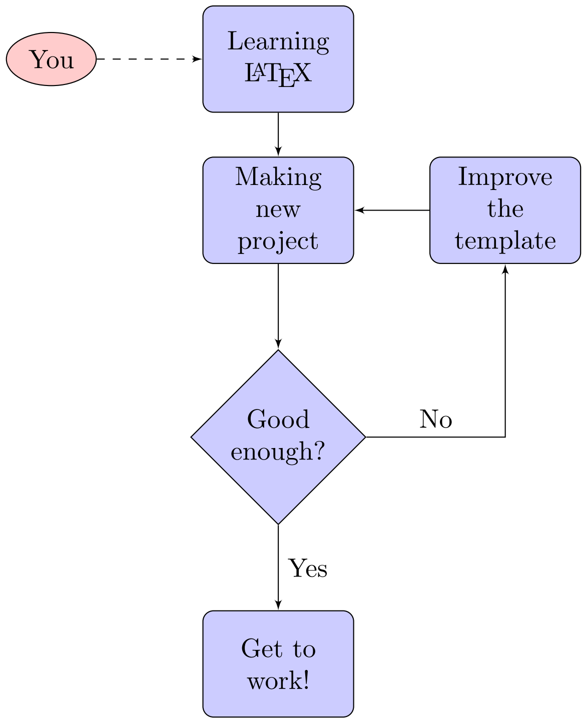


Figure 3: Example of generated flow chart

C Tables

Similarly to other markup languages, such as HTML, LaTeX documents are fundamentally one-dimensional, making two-dimensional concepts, such as tables, somewhat annoying to deal with. The following examples document ways to create and manage tables in LaTeX projects.

C.1 Basic

In the simplest case, tables can be created by manually typing out a set of LaTeX commands enclosed in a set of `\begin{table}...\end{table}` tags. Similarly to figures and sections, they can also be labeled and cross-referenced throughout the documents (see source for Table 1 for a commented example). Getting the formatting right can be a bit cumbersome, but there exists serveral websites[2] that can be used to generate the necessary markup from a simpler user interface.

Statistic	Agent57	R2D2 (bandit)
Mean	4766.25	5461.66
Median	1933.49	2357.92

Table 1: This is an example table.

[2]Latex table generator: `https://www.tablesgenerator.com`

C.2 Using Python

Alternatively, if the data you wish to visualize is already in python, the pandas package can be used to generate LaTeX markup as a string that can be copied into the LaTeX document.

```python
import pandas as pd
df = pd.DataFrame(table_data)
# Generates a latex table
tex1 = df.to_latex()
# Generates a latex table without the DataFrame's index as a column
tex2 = df.to_latex(index=False)
# Alternatively, save it to csv and use the method in the next section
df.to_csv('table_data.csv')
```

Note that the `to_latex` method will by default only generate the `\begin{tabular}...\end{tabular}` part of of the table. Additionally, sometimes it may not generate the exact format you might be looking for. However, a lot of this can be fixed with optional key-word arguments that are further explained in the pandas documentation (see `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to_latex.html`).

C.3 Using External Data

A third alternative is to keep all the table data in a separate file and read it into the document when it is compiled. This is arguably the most tidy way to generate tables since it separates the raw data from the visualization markup. The example in Table 2 is generated with the `csvsimple` package using data from "Tables/example_table.csv".

Statistics	Agent57	R2D2 (bandit)
Mean	4766.25	5461.66
Median	1933.49	2357.92

Table 2: Same data as Table 1, except generated from a csv file.

D Equations

LaTex is great for writing equations and mathematical symbols. To write an equation or expression, you must enter math mode. There are two forms of math modes; *inline* and *display*.

The *inline* mode can be activated by enclosing the expression in `$...$` symbols[3] and will placed inside the surrounding text. For instance, `$E = mc^2$` becomes $E = mc^2$. Inline mode is suited for short expressions, or for referring to variables or parts of larger expressions.

The *displayed* mode puts the mathematical expression on its own line, analogous to a figure or table. It can be activated with `\begin{equation}...\end{equation}`[4] and is generally more suited for larger expressions or equations you want to put extra emphasis on. Additionally, it allows the equation to be labeled and referenced throughout the document (see Equation 1).

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_{s'\sim\mathcal{T},a\sim\pi}\left[G_t|S_t = s\right] \\ &= \sum_{a,s'} \pi(a|s)\mathcal{T}(s'|s,a)\left[\mathcal{R}(s,a,s') + \gamma v_\pi(s')\right] \end{aligned} \tag{1}$$

[3]Inline math mode can also be activated with `\(...\)` or `\begin{math}...\end{math}`
[4]Displayed math mode can also be activated with `\[...\]`

Most fancy symbols and Greek letters are built into latex and can be written in math mode with a \ prefix. For instance, `\pi` becomes π. See `https://oeis.org/wiki/List_of_LaTeX_mathematical_symbols` for more information.

E Code Snippets

While algorithms can often be explained sufficiently with a few equations and a paragraph of text, a more succinct description can often be given by simply writing out the code or something resembling a working implementation. Here are a few examples of how this can be accomplished.

E.1 Pseudocode

It is common in scientific writing to not write out algorithms in any directly compilable or executable language, but instead use pseudocode. An advantage of this notation is that you are not locked into any language-specific syntax and may choose to write operations at any desired level of abstraction and even include standard equations. The example in Algorithm 1 illustrates how this can be done with the `algorithm` and `algpseudocode` packages.

Algorithm 1 Example Algorithm

```
procedure EUCLID($a, b$)                          ▷ The g.c.d. of a and b
    $r \leftarrow a \bmod b$
    while $r \neq 0$ do                           ▷ We have the answer if r is 0
        $a \leftarrow b$
        $b \leftarrow r$
        $r \leftarrow a \bmod b$
    end while
    return $b$                                    ▷ The gcd is b
end procedure
```

E.2 Conventional Code

In some situations, it might be more appropriate to disclose runnable code in an existing language. For instance, if a key part of your project involves a (relatively compact) implementation of an algorithm, you can include it in an appendix for enhanced portability and reproducibility. The example below uses the `minted` package to render a snippet of C++ code with conventional syntax highlighting.

```cpp
int main {
    // This is a comment
    std::cout << "Hello World from C++!" << std::endl;
    return 0;
}
```

E.3 Loading from File

Similarly to the table example in Section C.3, the LaTeX document can be kept tidier by loading data from an external file. The snippet below is loaded from "Code/HelloWorld.m" with the `minted` package.

```matlab
% This is a comment
disp("Hello World from MATLAB!");
disp("I am using the "tango" style to print this code in beautiful colors");
```