

Software for Mathematical Scientists and Educators

Joshua Maglione

January 4, 2024

Contents

1	Mathematical Typesetting and \LaTeX	2
1.1	How to get \LaTeX	2
1.2	Workflow and document structure	3
1.3	Source files	4
1.4	An introduction to \LaTeX syntax	4
1.5	Dynamic bibliographies and $\text{BIB}\TeX$	5
1.6	Mathematical symbols and formulae	6
1.7	\LaTeX Environments	6
1.8	Document layout	7
1.9	Some \LaTeX packages	7

Introduction

Communicating mathematics and performing long computations are both vitally important and challenging. Thankfully there is a wide selection of software to make these tasks more manageable. In this module, we will explore software used by everyday mathematicians and scientists. These include biologists, chemists, computer scientists, data scientists, financial analysts, educators, engineers, and physicists.

The goal is to build a foundation by using some of the most ubiquitous software in the field. This will help students throughout their career in and out of university. We will cover four topics in this module:

1. Mathematical Typesetting and \LaTeX ,
2. Python and Jupyter Notebooks,
3. Introduction to Programming,
4. Symbolic Computation and SageMath.

1 Mathematical Typesetting and \LaTeX

With advances in printing and typesetting, the question of how to produce high-quality mathematical symbols and texts is challenging. Without going through the history, we now have essentially two main styles of software to write mathematical formulae, diagrams, and images:

1. What-You-See-Is-What-You-Get (WYSIWYG) and
2. typesetting software (or write-format-preview style).

Software like Microsoft Word, Apple Pages, or LibreOffice Writer are WYSIWYG editors because you see and edit the document as a final product (regardless of whether or not it is the final product). This removes the user from having to remember commands for the document layout, and it has a lower barrier of entry, which is one of its strongest advantages. However, this is not the norm for professionals using many mathematical symbols, and the primary disadvantage to these kinds of software is their sluggish pace.

One of the first typesetting software for mathematics is \TeX —if not *the* first. It was written by Donald Knuth¹ in 1978 [3], and it is the cornerstone of the more modern software \LaTeX written by Leslie Lamport in 1986 [5]. Although \TeX is still used today, \LaTeX is far more popular. The primary disadvantage to these systems is the higher barrier to entry, but the over-powering advantage is the speed with which one can produce beautiful and high-quality mathematical symbols. Both \TeX and \LaTeX are free and open-source software.

As alluded to previously, these are typesetting software, so a user writes in a markup language, usually in a `tex`-file; then the user compiles the file, and a `pdf`-file is produced. There are other options for output, but this will be sufficient for us. Another asset is that one can import third party \LaTeX packages to perform more specialized tasks. We will become familiar with basic \LaTeX formatting and use some packages to help construct beautiful documents.

For web-based mathematical symbols, MathJax [2] is primarily used. However, there is a new, much faster, alternative called KaTeX [1]. Currently, KaTeX can only do a (proper) subset of what MathJax can do, but KaTeX does enough for everything I have needed for my website.

1.1 How to get \LaTeX

This will not fully cover how to install \LaTeX on your own machine, but hopefully it gives you enough information to help you. It is not that \LaTeX updates so frequently—in fact \LaTeX is still on its second edition (formatted as $\LaTeX 2_{\epsilon}$)—it is that packages tend to update or just need installation.

For Windows machines, I would recommend MiKTeX. This installs LaTeX and comes with a package manager to help with package installation. A similar version is available on Mac OS, and it is called MacTeX. Both MiKTeX and MacTeX

¹You can read Knuth's original memo describing \TeX . He called the memo the "Preliminary preliminary description of \TeX " [4].

have graphical user interfaces. For Linux systems, TeX Live is what I recommend, and it usually comes pre-installed. It has its own package manager invoked by the command `tlmgr`.

We will primarily use Overleaf in this module. Overleaf is a website that enables users to interface with L^AT_EX through cloud-based services. It uses a “freeium” model, so that everyone can use the basic features, which will be sufficient for our module. The major advantage is that one does not have to worry about installing and package management; all of this is done cloud-side. Moreover Overleaf simplifies the workflow slightly by allowing for instant compilation.

1.2 Workflow and document structure

The basic workflow is perhaps only a little more complicated than how it might be for WYSIWYG software. Here is the basic workflow.

1. Create a tex-file and write L^AT_EX markup.
2. Compile the tex-file with the command `pdflatex`.
3. Sometimes errors are raised and need to be addressed. If no errors arise, then a pdf-file is created (or overwritten).
4. View and review the output pdf-file.

With the exception of the initial creation of the file, all of these steps are repeated often. How often? That depends on you and your situation. I would recommend that, when addressing bugs or typos, that one compile often and review carefully what has changed.

The basic format of a L^AT_EX document is simple. There are three commands that must be present, and often one tries to adhere to some logical structure due to collaborations or just to readability over longer periods of time. The first command that appears in a functioning tex-file is the following:

```
\documentclass[<options>]{<style>}
```

where `<style>` is replaced with the name of the desired document class and `<options>` is replaced with optional parameters for the specific document class. For example, this current pdf document was built using

```
\documentclass[a4paper, 12pt]{article}
```

Some other popular document classes, besides `article`, are

- `beamer` : for slides,
- `book` : for books,
- `letter` : for letters,
- `standalone` : often used with the package `TikZ` for stand alone pictures.

There are also AMS-inspired article and book classes: `amsart` and `amsbook` and KOMA-script versions: `scrartcl`, `scrbook`, `scrlettr2`, and `scrreprt`. The other two required commands are

```
\begin{document}  
\end{document}
```

which encapsulate the main body of the tex-file. Figure 1.1 shows the basic layout of a tex-file.

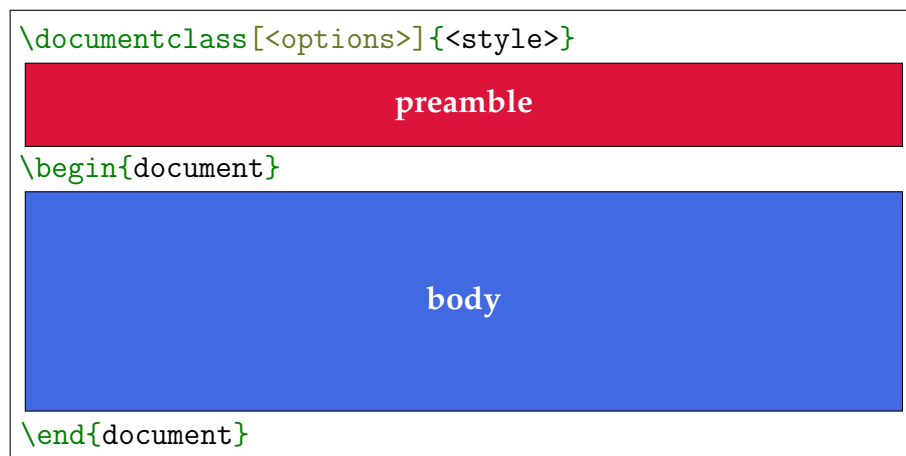


Figure 1.1: Basic structure of a tex file.

1.3 Source files

\LaTeX *source files* are the necessary files needed to compile a tex file. In particular, a tex file need not be self contained; there are many reasons why this might be the case. For example, one might embed graphics in the form of jpeg or png files, or one might use \BibTeX to dynamically format their bibliography—more on this in Section 1.5.

In the process of compiling a tex file to a pdf file, \LaTeX produces several auxiliary files. These serve specific purposes, but their particular uses are outside of the scope we will explore. Because so many additional files are created in the compilation process, it is usually preferred to build a directory for each \LaTeX project. For example, the file names in the directory for these lecture notes can be seen in Figure 1.2.

The only files necessary to compile in Figure 1.2 are `LectureNotes.tex` and `bibliography.bib`. The rest of the files are produced by the compilation process.

1.4 An introduction to \LaTeX syntax

We will not cover all of the \LaTeX syntax here. When we discuss mathematical symbols and formulae in Section 1.6, we will cover more. There are some symbols that \LaTeX redefines. For example, `%` indicates that the rest of the line should be ignored by the compiler:

LectureNotes	
bibliography.bib	LectureNotes.log
LectureNotes.aux	LectureNotes.out
LectureNotes.bbl	LectureNotes.pdf
LectureNotes.blg	LectureNotes.tex
LectureNotes.fdb_latexmk	LectureNotes.toc
LectureNotes.fls	LectureNotes.synctex.gz

Figure 1.2: The files in the LectureNotes directory.

```
% This is a comment and will be ignored by the
% LaTeX compiler.
```

Commands in \LaTeX start with the `\` symbol. We have already seen examples of this with the following commands:

```
\documentclass[a4paper, 12pt]{article}
\begin{document}
\end{document}
```

Arguments are input using the `{` and `}` symbols, and multiple arguments would require multiple sets of `{` and `}`. Optional parameters are input using the `[` and `]` symbols, but multiple optional parameters are listed within the single use of `[` and `]` separated by commas. Not every command requires input arguments; for example, to format LaTeX as \LaTeX , one uses the command `\LaTeX`.

1.5 Dynamic bibliographies and \BIBTeX

Another major advantage to \LaTeX is its ability to dynamically format bibliographies and references. There are a number of different ways to format bibliographies in \LaTeX , but we will discuss specifically \BIBTeX , written by Oren Patashnik and Leslie Lamport in the 1980s.

Have you ever written a document, and after having gotten halfway through, say, you decide to add another reference? Depending on the style adopted in your document, this might have caused you lots of additional work. Not one would you have to potentially re-order your bibliography, but you might have to change several references to account for this. Having a dynamically built bibliography avoids all of this pain.

To use \BIBTeX , one simply includes two lines wherever the references should go—usually at the end of the document:

```
\bibliography{<bib file name>}
\bibliographystyle{<style>}
```

Thus, in order to use \BIBTeX one needs an additional source file: a bib file. A bib file collects bibliography entries that \BIBTeX can read. Figure 1.3 provides an example.

```

@book{Macdonald1998,
  title={Symmetric functions and Hall polynomials},
  author={Macdonald, Ian Grant},
  year={1998},
  publisher={Oxford university press}
}

```

Figure 1.3: A sample BIB_TE_X entry.

In the first line of Figure 1.3, `@book` tells BIB_TE_X that the bibliography entry is a book. The braces indicate the block of code designated for this particular entry. And the `Macdonald1998` is the *tag*—more on this soon. The lines in between the braces give bibliographical information. As the example illustrates: we have the book’s title, author, year of publication, and publisher. There are more potential entries one could include as well. The entry will be formatted relative to the specified style in the output pdf file.

When one wants to cite a particular reference in the body of their tex file, one simply writes

```
\cite[<options>]{<tag>}
```

and a citation for that particular entry (relative to the tag) matching the given style is produced. Using the example in Figure 1.3, we might write the following sentence in the body of our tex file:

```
By \cite[Theorem 3.14]{Macdonald1998}, the proof is complete.
```

The bib file does not need to be ordered. BIB_TE_X will take care of ordering all of the bibliography entries properly. Therefore, the bib file can simply be a dump for bibliography entries. All a user needs to know are the relevant tags for the references to cite.

Using BIB_TE_X modifies the basic workflow described in Section 1.2. After running `pdflatex` one needs to run `bibtex` twice, and then `pdflatex` again. Overleaf and other similar programs can run all of this with one button, so we will not dwell on this.

Remark 1.1 (Warning). BIB_TE_X can be challenging for newcomers to work with for many reasons. Some of the pain is alleviated by Overleaf and other programs, but this still does not make it easy.

1.6 Mathematical symbols and formulae

1.7 L_AT_EX Environments

One of the general blueprints of L_AT_EX markup is an *environment*. A L_AT_EX environment has the following basic structure:

```
\begin{<environment>}      % Explicit start of environment
%
% Content
%
\end{<environment>}        % Explicit end of environment
```

One could say that a tex file is essentially a \LaTeX environment using document. We will see many examples of these later.

1.8 Document layout

1.9 Some \LaTeX packages

References

- [1] Khan Academy. KaTeX. katex.org, 2024.
- [2] MathJax Consortium. MathJax. mathjax.org, 2024.
- [3] Susan D’Agostino. The computer scientist who can’t stop telling stories. *Quanta Magazine*, 2020. Retrieved on [03-Jan-2024](#).
- [4] Donald Knuth. Preliminary preliminary description of TEX. [TEXDR.AFT](#), 1977.
- [5] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley Longman Publishing Co., Inc., USA, 1989.