Software for Mathematical Scientists and Educators

Joshua Maglione

January 4, 2024

Contents

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 remove 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 Knuth[1] 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 not that LATEX updates so frequently—in fact LATEX is still on its second edition (formatted as LATEX 2_ε)—it is that packages tend to update or just need installation.

For Windows machines, I would recommend MiKTeX. This install 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 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`.

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

1. Create a `tex`-file and write LaTeX 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.

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.