CAAM 519: Computational Science I

Module 3: Typesetting software

As scientists, coming up with brilliant new ideas is only half the battle. You also have to be able to communicate your ideas, and to convince others that your ideas are worthwhile. While its out of scope to teach you how to how to do research or how to sell your ideas effectively, we can teach a small but important piece of the story: how to typeset your written work. While this may seem trivial, it's in fact a very important and often subtle skill to learn, particularly in the mathematical sciences which are heavy in notation, unusual symbols, and complex strings of equations.

This module will provide an overview of some of your options for typesetting your scientific writing. In particular, we will focus on LaTeX, which is the de-facto standard in mathematics and adjacent communities.

Word processors

A word processor is a program that allows you to manipulate text. You have almost certainly used a word processor before; most likely, it was Microsoft Word. Word processors are great for many tasks, as their widespread use bears out. Word processors operate under the "what you see is what you get" principle, and so are quite user-friendly with a low barrier to entry. However, their support for mathematical typesetting is fairly anemic and not much good for anything more complex than a few Greek characters embedded in text.

Nevertheless, you'll note that the lecture notes for this class are written in Google Docs. This is because we haven't seen much mathematics (yet, at least), and Google Docs has fairly good support for typesetting code. Moreover, the barrier to entry is almost zero, and free online hosting for Google Docs makes it easy to share and collaborate on your work. Therefore, Google Docs can make a handy tool, for example, for writing *design docs*: a description of a complex software project or feature you write before starting to clearly articulate your design decisions to yourself and others.

This is all just to say that there is no one right tool for every job. Though LaTeX is incredibly powerful, it can be cumbersome and comes with a nontrivial activation cost, and oftentimes you may be better suited using other tools instead.

Markdown

Markdown is a *markup*[1] language that is an extremely lightweight of converting plaintext to HTML code. It is not one particular program, like Microsoft Word. Rather, it is closer to a programming language, specialized for typesetting formatted text, where the code looks strikingly like plain text. By using just a few extra characters, you can encode formatting specifications directly in your text: things like headers, italics, hyperlinks, and so on.

As mentioned, Markdown is extremely lightweight, and text written in Markdown reads like plaintext. While the math formatting is nonexistent, Markdown is really great for other tasks: for example, documenting your software.

To start, it's easiest to list the basic functionality (modulo weird corner cases that rarely arise):

- Headers:

```
# First level header
## Second level header
### Third level header
```

- Emphasis:

```
*italicized*
**bolded**
```

- Lists:

```
* First item
* Second item
```

- Ordered lists:

```
1. First item
2. Second item
```

- Hyperlinks:

```
[Markdown
spec](https://https://daringfireball.net/projects/markdown/basics)
```

- Code:

````
Inline with ``ls -al``
Block of code (with the "sh" denotes the language for code highlighting)
```sh
cd ~
mkdir markdown
````

[1] A markup language is one which uses tags to define elements such as formatting directly within plaintext. For example, HTML is a markup language.
````

```
```
```

- Blockquote:

```
> First line of block quote...
> ...second line here, and so on.
```

So once you have written a Markdown file, how do you properly view it? For that you need a program to render it. Since Markdown is really just a text-to-html conversion spec, there are numerous Markdown renderers at your disposal. For example, hosting sites like Github and a number of text editors (e.g. Atom), have packages that support in-window rendering. Furthermore, there are a number of websites (easily found via a Google search) that will give you the raw html output.

LaTeX

LaTeX is the gold standard for typesetting mathematics, and is the lingua franca for academics in mathematics and related fields in the sciences and engineering. LaTeX is built on top of the low-level typesetting software TeX, developed by Donald Knuth for his storied “The Art of Computer Programming” series. As such, LaTeX is really just a collection of macros to make the language more user-friendly. There are a number of alternatives (such as LuaTex, XeLaTeX, ConTeXt, etc.) that serve the same role and offer more “modern” features like Unicode support. However, none has reached the level of adoption anywhere close to LaTeX.

LaTeX is a markup language that you write as plaintext, and typically gets compiled to the desired output format (e.g. a pdf file). Unsurprisingly, you can compile your documents directly from the terminal, which is how we will start. To install the LaTeX on your Linux machine[2], run

```
> sudo apt install texlive
```

This installs a number of programs and libraries, but for the main user-facing program you will interact with is `pdflatex`, which compiles LaTeX code and renders it to a pdf.

A first latex document

Let’s start with an extremely simple LaTeX document. Copy the following into your favorite text editor, and save it as `my-first-latex.tex`:

[2] On a Mac machine, you can install using [MacTex](), and on Windows using [MikTeX]().
```

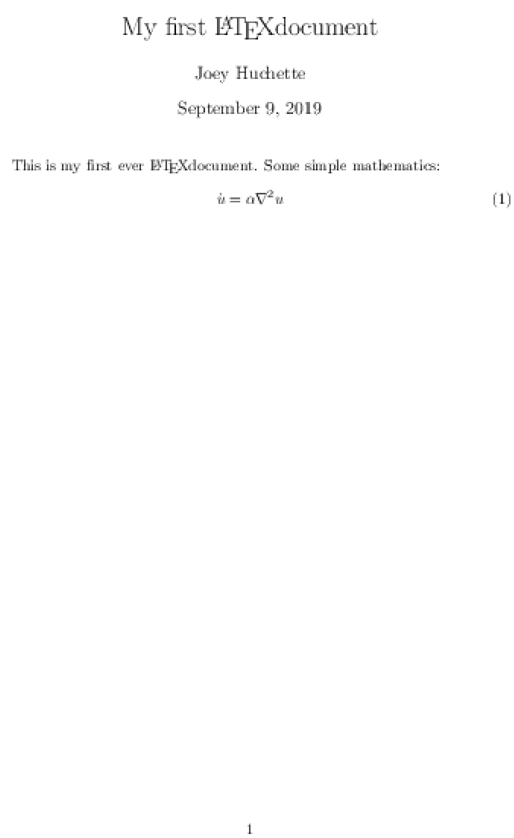
```latex
\documentclass{article}
\usepackage{amsmath}
\title{My first \LaTeX document} % Backslash to insert a special character
\author{Joey Huchette}
\begin{document} % We content of the document is specified in this block
\maketitle
This is my first ever \LaTeX document. The heat equation is
\begin{equation}
      \dot{u} = \alpha \nabla^2 u.
      \end{equation}
\end{document}
```

Then, to compile this document, simply run

```
> pdflatex my-first-latex.tex
This is pdfTeX, Version 3.14159265-2.6-1.40.20 (TeX Live 2019) (preloaded
format=pdflatex)
 restricted \write18 enabled.
entering extended mode
(./my-first-latex.tex
LaTeX2e <2018-12-01>
(/usr/local/texlive/2019/texmf-dist/tex/latex/base/article.cls
Document Class: article 2018/09/03 v1.4i Standard LaTeX document class
(/usr/local/texlive/2019/texmf-dist/tex/latex/base/size10.clo))
(/usr/local/texlive/2019/texmf-dist/tex/latex/amsmath/amsmath.sty
For additional information on amsmath, use the `?' option.
(/usr/local/texlive/2019/texmf-dist/tex/latex/amsmath/amstext.sty
(/usr/local/texlive/2019/texmf-dist/tex/latex/amsmath/amsgen.sty))
(/usr/local/texlive/2019/texmf-dist/tex/latex/amsmath/amsbsy.sty)
(/usr/local/texlive/2019/texmf-dist/tex/latex/amsmath/amsopn.sty))
No file my-first-latex.aux.
[1{/usr/local/texlive/2019/texmf-var/fonts/map/pdftex/updmap/pdftex.map}]
(./my-first-latex.aux)
)</usr/local/texlive/2019/texmf-dist/fonts/type1/public/
amsfonts/cm/cmmi10.pfb></usr/local/texlive/2019/texmf-dist/fonts/type1/publ
ic/a
msfonts/cm/cmr10.pfb></usr/local/texlive/2019/texmf-dist/fonts/type1/public
/ams
fonts/cm/cmr12.pfb></usr/local/texlive/2019/texmf-dist/fonts/type1/public/a
msfo
nts/cm/cmr17.pfb></usr/local/texlive/2019/texmf-dist/fonts/type1/public/ams
```

```
font
s/cm/cmr7.pfb></usr/local/texlive/2019/texmf-dist/fonts/type1/public/amsfon
ts/c
m/cmsy10.pfb>
Output written on my-first-latex.pdf (1 page, 59065 bytes).
Transcript written on my-first-latex.log.
```

This will print a bunch of logging information to STDOUT, and if you typed in everything correctly, you will get the rendered document saved as `my-first-latex.pdf`. It should look something like this:



My first LaTeXdocument

Joey Huchette

September 9, 2019

This is my first ever LaTeXdocument. Some simple mathematics:

$$\dot{u} = \alpha \nabla^2 u \tag{1}$$

1

Note also that the compilation process creates auxiliary logging files:

```
> ls
my-first-latex.aux my-first-latex.log my-first-latex.pdf my-first-latex.tex
```

Let's break the LaTeX code down line-by-line. First, we start with the *preamble*.

First, we tell LaTeX which type of document we would like to write. There are a number of built-in document classes (book, letter, beamer, etc.), but "article" is a typical choice for a written report. The document class specifies helpful default configurations (e.g. page size and layout). Often, journals will have custom article classes that you must use before submitting papers.

Next, we see how to load LaTeX packages. A standard TeX distribution will come with a large number of libraries for esoteric tasks, but you must explicitly load them to use them. Here, we load the amsmath package, which is the standard library for mathematical typesetting in LaTeX. You will likely be loading this package in every LaTeX document you write.

On the third line, we specify a title for our document. Note the special escape charAt this point, let's observe that we have seen a large number of backslashes (`\`) so far in our LaTeX document. LaTeX uses backslashes as an escape character, meant to signify that the following string of characters is meant to be interpreted as a markup command, rather than plain text. Escaped commands will start with a backslash, followed by a single word specifying the command. Required arguments to the command are passed inside curly brackets (`{}`), and optional ones through square brackets (`[]`). On this line we also see a comment, which is started with a `%` character. On the following line, we specify an author for the document.

Next, we begin the document block. This ends the preamble, and specifies the contents of the document. On the first line inside the document block, we issue the `\maketitle` command to print the title at the top of the document. Beyond this, the contents of the document are some plain text, and then a single line of mathematics.

To format the math, we use the equation environment, which allows us to enter a single line equation. The contents inside the equation environment are interpreted in math mode, meaning that we have access to a host of handy new commands for math formatting: e.g., Greek letters, exponents, and annotations.

More on math mode

There are a number of ways to format math in LaTeX. You can typeset math in-line, interspersed in ordinary text, using the `$$` delimiters:

```
The heat equation, $\dot{u} = \alpha \nabla^2 u$, is a fundamental partial differential equation.
```

The heat equation, $\dot{u} = \alpha \nabla^2 u$, is a fundamental partial differential equation.

We have already seen the equation environment above. You will also find frequent use for the gather and align environments, which allow you to typeset multiple lines of equations. In both, you denote a line break with a double backslash. Additionally, the align environment allows you to use the & character to specify alignment among different rows in the environment, which is often quite handy:

```
\begin{align*}
      \pi &= 16\tan^{-1}\left( \frac{1}{5} \right) - 4\tan^{-1}\left(
\frac{1}{239} \right) \\
      &= 4\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1} \\
      &= \sqrt{6\left( 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} +
\cdots \right)}.
\end{align*}
```

$$
\begin{aligned}
\pi &= 16\tan^{-1}\left(\frac{1}{5} \right) - 4\tan^{-1}\left(\frac{1}{239} \right) \\
&= 4\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1} \\
&= \sqrt{6\left(1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots \right)}.
\end{aligned}
$$

A couple additional things of note here. First, note the * appearing in the environment command. This is a common LaTeX idiom which suppresses any labels. Second, note that if you want to specify sub/super-scripts with more than one character, you must use the curly bracket delimiters. Third, we can resize certain delimiters (e.g. parentheses), based on the size of its contents, using the `\left` and `\right` delimiters.

If you want to explicitly label an equation (or only one line in an align or gather), it is possible to label each line separately, and refer to them elsewhere in the text:

```
\begin{align}
```

```
    \pi &= 16\tan^{-1}\left( \frac{1}{5} \right) - 4\tan^{-1}\left(
\frac{1}{239} \right) \\
    &= 4\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1} \\
    &= \sqrt{6\left( 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} +
\cdots \right)} \label{eqn:euler}.
\end{align}
The representation for $\pi$ given in \eqref{eqn:euler} was obtained by
Euler as a special case of the Riemann zeta function.
```

$$\pi = 16\tan^{-1}\left(\frac{1}{5}\right) - 4\tan^{-1}\left(\frac{1}{239}\right) \tag{1}$$

$$= 4\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{2k-1} \tag{2}$$

$$= \sqrt{6\left(1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\cdots\right)}. \tag{3}$$

The representation for π given in (3) was obtained by Euler as a special case of the Riemann zeta function.

You can also label an entire environment block of math, and refer to this entire block, using the subequations environment:

```
\begin{subequations} \label{eqns:pi}
\begin{align}
    \pi &= 16\tan^{-1}\left( \frac{1}{5} \right) - 4\tan^{-1}\left(
\frac{1}{239} \right) \\
    &= 4\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1} \\
    &= \sqrt{6\left( 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} +
\cdots \right)}.
\end{align}
\end{subequations}
A few of the many elegant expressions for the value of $\pi$ are given in
\eqref{eqns:pi}.
```

$$
\begin{aligned}
\pi &= 16 \tan^{-1}\left(\frac{1}{5}\right) - 4 \tan^{-1}\left(\frac{1}{239}\right) && \text{(1a)} \\
&= 4 \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1} && \text{(1b)} \\
&= \sqrt{6\left(1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots\right)}. && \text{(1c)}
\end{aligned}
$$

A few of the many elegant expressions for the value of π are given in (1).

Figures and tables

Inserting figures into LaTeX document is a bit more involved than in a word processor like Microsoft Word, but can still be accomplished with just a few commands. First, make sure you load the appropriate package in the preamble with `\usepackage{graphicx}`. Then, you can add a figure to your document with:

```
Some text preceding Figure~\ref{fig:rice-logo}.

\begin{figure}[htbp]
      \includegraphics[width=0.8\linewidth]{rice-logo.jpg}
      \caption{Foo bar}
      \label{fig:rice-logo}
\end{figure}

Some text following Figure~\ref{fig:rice-logo}.
```

Some text preceding Figure 1.

Figure 1: Foo bar

Some text following Figure 1.

At first you might be surprised by the placement of the image. In particular, depending on the rest of the contents of your document, the figure might appear on a completely different page than the text preceding or following it in the markup. In fact, this is what the `[htbp]` options help control: it gives a ranking of where you would like to place the image: “here”, top of page, bottom of page, or top of next page (respectively).

You can specify tables using the `tabular` environment:

```
\begin{table}[htbp]
      \begin{tabular}{r|l|cc}
            1 & 2 & 3 & 4 \\ \hline
            5 & 6 & 7 & 8 \\
            9 & 10 & 11 & 12 \\
      \end{tabular}
\end{table}
```

1	2	3	4
5	6	7	8
9	10	11	12

Here the options to `tabular` specify the orientation of each column in the table: the first column is right justified, the second left justified, and the last two are centered. Additionally, there is a vertical line drawn between the first and second, and second and third, columns.

We enter the contents of the table, distinguishing between cells using &, and breaking lines with double backslashes. Moreover, we can draw an explicit horizontal line with the `hline` command.

References

There are a number of handy tools available in LaTeX to handle references. The simplest way is to add references is the following:

```
\documentclass{article}
\begin{document}
The suggested readings for this course cover Latex~\cite{latex},
C~\cite{c}, and C++~\cite{cpp}.
\begin{thebibliography}{3}
\bibitem{latex}
Michel Goossens, Frank Mittelbach, and Alexander Samarin.
\textit{The \LaTeX\ Companion}.
Addison-Wesley, 1993.
```

```
      \bibitem{c}
            Brian Kernighan and Dennis Ritchie.
            \textit{The C Programming Language}.
            Prentice Hall, 1988.
      \bibitem{cpp}
            Bjarne Stoustrup.
            \textit{The C++ Programming Language}.
            Addison-Wesley, 2013.
\end{thebibliography}
\end{document}
```

The suggested readings for this course cover Latex [1], C [2], and C++ [3].

References

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, 1993.

[2] Brian Kernighan and Dennis Ritchie. *The C Programming Language*. Prentice Hall, 1988.

[3] Bjarne Stoustrup. *The C++ Programming Language*. Addison-Wesley, 2013.

Other handy tools

Bibtex

While the `thebibliography` environment is more handy than manually tracking the references, it is not particularly scalable to larger projects, or to scenarios where you are handling multiple projects with overlapping references. Therefore, I highly recommend you use a bibliography management tool like BibTeX. Essentially, BibTeX stores all the bibliographic information in a separate file, and there are a number of handy applications to manage and generate these .bib files. See, for example, BibDesk or Mendeley.

Once you have generated the `.bib` file, using it in your LaTeX project is as easy as copying the bibliography file (say, `references.bib`) to the build directory, and adding the following lines at the end of your document:

```
\bibliographystyle{siam} % Can change style based on preferences and
journal standards
```

```
\bibliography{references}
```

Latexit

Sometimes it's useful to generate snippets of LaTeX output, rather than go through the process of generating an entire document and snipping it manually. For example, you might want just one line of equations to include in a Powerpoint presentation. The handy tool LaTeXiT does this for you. It comes included with many standard TeX installations.

Git or Overleaf for project collaboration

Oftentimes you will need to collaborate on LaTeX projects with others. Since LaTeX is a plaintext markup language, it is a natural fit for a version control system such as Git.

Alternatively, Overleaf is a product for hosting and collaborating LaTeX documents online (and is free for academics). Think of it like Github, but for LaTeX projects, with a decent real time collaboration feature a la Google Docs.