Advanced LaTeX Syllabus

OSU Math Club
mathclub@math.oregonstate.edu

April 2020

Abstract

The full functionality of LaTeXin a document! This document is a practitioners guide to using LaTeX, the features you should know to improve your document appearance. The table of contents is on page 9. Included are fundamentals of what LaTeX is and what is does, how to build a document, page layout, custom commands/macros, math environments, theorem environments (for theorems, lemmas, remarks, etc.), using images, references, and nesting documents (including files in your file), code snippets, bibliographies/citations with bibtex, SI units, and embedding external references.

1 File types

When working with LaTeX, you'll often run across files beyond .tex files. These are

.sty Style files: a text file implementing packages

.log Log file: a file produced by the latex compiler that describes everything that the compiler did, including any reformatting, package usage, or errors generated. The end of every (successful) log file contains a description of how big the document was.

.aux Auxiliary files: A temporary file generated for storing document details, includes build information: sectioning, page numbering, references, where to insert bibliographic reference numbering, etc. Aux files are passed between pdflatex and bibtex/biber/external tools and overwritten at the end of pdflatex.

.cls Class files: A file specified by the argument to \documentclass. Latex comes with a few: the most popular are article, beamer (slides), and memoir (book-like). Class files can be provided by packages or written as standalone files.

2 The build sequence

What's LaTeX doing? LaTeX is built on a mishmash of WEB2E (a derivative of java) styled after pascal, with a C interpreter. The most common executable is not actually LaTeX which returns a type of graphic file called postscript, but pdflatex which returns a pdf. To build a document, pdflatex is called with a few arguments, the details of which are not terribly important to you. The *Build Sequence* is the sequence of commands to properly build your document. LaTeX is an executed language, it doesn't handle referencing with extra files (like itself) well. To overcome this, it produces a auxilliary file (.aux) which allows information to be stored between calls. When pdflatex is called, it looks for a matching .aux file and passes that into the the compiler.

To overcome this, build sequences for complicated documents require multiple pdflatex calls. The standard build sequence is

- pdflatex (to generate labels)
- biber or bibtex (for citations)
- pdflatex (to generate reference requests)
- pdflatex (to match labels and citations to references)

Most IDEs will automate this process for you with a build function. TeXstudio has a build-and-view (f5) and a compile (f6). If you're not working with referencing, both give the same result. If you're working with references but aren't editing them, you can do a compiler call after a build call. Overleaf hides this technical detail from you and is prepared to run a build cycle every time you push refresh.

3 Fancy headers and footers

3.1 Formatting

The package fancyhr ("Fancy header“) provides tools to modify the header and footer.

- First import the package with \usepackagefancyhdr
- Set the page style to fancy with \pagestyle{fancy}
- This will add a section header to each non-title page and a line to separate it from the main text.
- To modify the header or footer, first identify the command.
  - Position: left, center, or right as l, c, or r
  - header or footer as head or foot
  - Examples: \lhead, \rfoot, \cfoot, \chead

More examples on Overleaf.

3.2 Page numbering

to display Page X of Y:

- Import the package lastpage.
- Reference the current page with \thepage
- Add "2of \pageref{LastPage}" within a fancyhdr section call (\cfoot or \lfoot is popular).

3.3 Timestamp your document

Warning: timestamps with Overleaf occur against their system clock which may not reflect your machine clock or physical time zone.
Timestamps are useful when emailing documents to collaborators when not using a version control system.

- Import the datetime package.
- \currenttime{} (from datetime) provides the current time
- \today provides the date

```
\rfoot{Compiled \currenttime{} on \today}
```

Example:
Compiled 18:33 on Tuesday 28$^{\text{th}}$ April, 2020
Default is 24 hour time behavior.

3.4 Alternate way to include timestamps

The datetime2 package provides support for this.

- Import datetime2.
- Access a timestamp with \DTMnow, will be formatted by system call.
- datetime and datetime2 conflict, so the latter package imported will be used.

3.5 Exercises

- Make a simple latex document with fancyheader. Place your name in the top right, date in the center, “Homework 4” in the top left, and “Math Club” in the bottom right.
- Add a timestamp to a corner using datetime. Modify the timestamp to reflect only the date.
- Update your document to use datetime2

4 Custom commands and Macros

4.1 New commands

Useful for “insert this chunk of text on a keyword call”.

- new macros/commands follow the formula \newcommand{\Call }{Stuff to Insert }
- Then call the macro with \Call.
- Example: Blackboard notation: \newcommand{\R}{\mathbb{R}}
- Custom commands can accept arguments! Use [n] to add n arguments. Access them in order with #number (#1, #2, . . . , #n)
- More examples: (My personal list, divided by application area)

```
% For derivatives
\newcommand{\deriv}[2]{\frac{\mathrm{d}#1}{\mathrm{d}#2}}

% For partial derivatives
\newcommand{\pderiv}[2]{\frac{\partial #1}{\partial #2} }

% Integral dx
\newcommand{\dx}{\mathrm{d}x}
\newcommand{\dd}{\mathrm{d}}

% Alias for the Solution section header
\newcommand{\solution}{\textbf{\large Solution} \\}

% Probability commands: Expectation, Variance, Covariance, Bias
\newcommand{\E}{\mathrm{E}}
\newcommand{\Var}{\mathrm{Var}}
\newcommand{\Cov}{\mathrm{Cov}}
\newcommand{\Bias}{\mathrm{Bias}}

% Real analysis commands:
\newcommand{\limn}{$\lim_{n \to \infty}$}
\newcommand{\infx}{ \text{inf}_{x \in [x_{i-1}, x_i]} }
\newcommand{\supx}{ \text{sup}_{x \in [x_{i-1}, x_i]} }
\newcommand{\ingab}{$[a,b]$}
\newcommand{\limnn}{\ensuremath{\lim_{n \to \infty}}}
\newcommand{\R}{\ensuremath{\mathbb{R}}}
\newcommand{\N}{\ensuremath{\mathbb{N}}}
\newcommand{\Z}{\ensuremath{\mathbb{Z}}}
\newcommand{\C}{\ensuremath{\mathbb{C}}}
\newcommand{\inv}{\ensuremath{^{-1}}}
%Abstract Algebra Commands
\newcommand{\MR}{\mathbb{M}_2(\mathbb{R}}
```

4.2 Redefining calls

Some packages will set bindings. You can find these with a complier error when you try and bind a macro to it.

- Use renewcommand in place of newcommand.
- If you want to save the old command (eg you need it elsewhere), use

  ```
  \let \newbinding\oldcommand
  ```

5 The many Math Environments of LaTeX 2ε

We'll add horizontal lines (\hrule) to emphasize spacings.

$... $ inline math mode. Good for "If $x \in Z$, then $f(x) = \ldots$.

$$... $$ A tex (the base application for LaTeX) primitive method for entering math mode. Don't use this, mentioned for completeness.

\[\ ] Shorthand method to enter display mode, a classic.

Displaymath The environment based way to get into display mode. Works like \[\] but more to type.

equation and equation* A way to get numbered equations, adds a bit of vertical space compared to display mode, a better version is available with amsmath. Use * to suppress numbering.

$$f(x) = \int_0^1 \sum_{n=0}^{\infty} a_n(x)\mathrm{d}x \tag{1}$$

align and align* Used for vertically aligning & characters. Useful for systems of equations and demonstrating work. Use * to suppress numbering. Adds space before math.

$$\begin{align} f(x) &= \int_0^1 x^n \mathrm{d}x \tag{2}\\ &= \frac{1}{n+1}1^{n+1} - 0 \tag{3}\\ &= \frac{1}{n+1} \tag{4} \end{align}$$

aligned Like align*, but without the vertical space above, but does not enter math mode for you. Less space from previous line.

$$\begin{aligned} f(x) &= \int_0^1 x^n \mathrm{d}x \\ &= \frac{1}{n+1}1^{n+1} - 0 \\ &= \frac{1}{n+1} \end{aligned}$$

eqnarray* Like align, but worse with horizontal spacing. Don't use this.

gather and gather* Provided by amsmath, center justifies all arguments. Use a \\

$$\begin{gather} a = b + 1 \tag{5}\\ b = a - 1 \tag{6}\\ a - b - 1 = 0 \implies q! \tag{7} \end{gather}$$

6 Theorems, lemmas, and remarks

Accessible through amsthrm (AMS theorems). You can define axioms, theorems, lemmas, corollaries, remarks, examples, definitions, conjectures, or custom names. Proof environments are also accessible.

1. Add \newtheorem{environment name }{Displayed Name}. Numbering for multiple instances occurs, but does not support other types, so you get Definition 1, Definition 2, Lemma 1, lemma 2, corollary 1, lemma 3.

2. A good naming convention to follow (to be consistent with standard latex usage) is to use lower case for environment names and Capitals for Display Names.

3. If you want numbering (eg def 1, lemma 2, def 3, corollary 4), we need to modify the declaration.

   ```
   \newtheorem{theorem}{Theorem}
   \newtheorem{remark}[theorem]{Remark}
   \newtheorem{axiom}[theorem]{Axiom}
   ```

4. To set the numbering into a section (so def 1 is def 1.1) and reset on section incrementing, change the initial declaration to

   ```
   \newtheorem{theorem}{Theorem}[section]
   \newtheorem{remark}[theorem]{Remark}
   \newtheorem{axiom}[theorem]{Axiom}
   ```

   The convention here of placing the [] after the {} is contrary to standard use, to avoid confusion with the reference to an environment for numbering.

5. Call an environment as

   ```
   \begin{theorem}[My named theorem]
   If 0 is a number, then \pi is awesome.
   \end{theorem}
   ```

6. You might want to follow your named environment with a proof.

   ```
   \begin{proof}
   The result follows from the imaginary Euler-Dedekind-Maruyama-Gauss-Newton
   -Seidel-Laplace-Strogatz-Tau-Ramanujan-Galois-Cantor-Cauchy theorem.
   \end{proof}
   ```

   Proof. The result follows from the Euler-Dedekind-Maruyama-Gauss-Newton-Seidel-Laplace-Strogatz-Tau-Ramanujan-Galois-Cantor-Cauchy theorem. □

The documentation for amsthrm is quite readable.

7 Images

Image support is provided by the graphicx package. The graphic and graphics package are depreciated and should not be used.

1. Import the graphicx package.

2. To include an image, use \includegraphics[]path/to/file

3. path/to/file supports relative and absolute imports. Relative imports are more stable than absolute imports (which should be avoided). The specific style is dependent on your operating system (windows vs linux style file paths).
4. For Windows: the \will convert your folder names to commands. Use / instead.
5. Graphicx supports literals for filepath names and can suppress wildcard searches. If you have Images/img entered as the filename, and your directory Images has img1 and img2, it will use the first one it finds.
6. This search is quite useful! You can often drop file format suffixes from the file listings. If you have a date stamp in image file names, you can ignore the last few digits (be careful with this though).

Vector graphics are preferred for latex documents (where images are stored as collections of vectors, rather than pixels) as opposed to raster (pixel maps) images. Supported file formats:

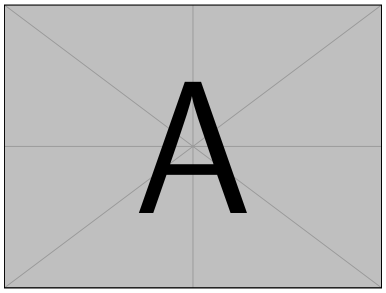
- EPS (preferred) - encapsulated post scripts. LaTeX includes a utility called eps2pdf which handles eps files beautifully. Most graphics outputting technical programs have an option to output an eps file.
- PNG - portable network graphics. Very popular for websites,
- JPEG (avoid) - not recommended for scientific applications due to lossy compression, good for websites due to high compression. Requires jpeg2eps converter.

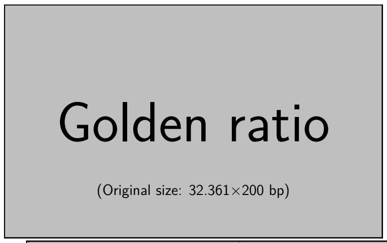
Unsupported file formats:

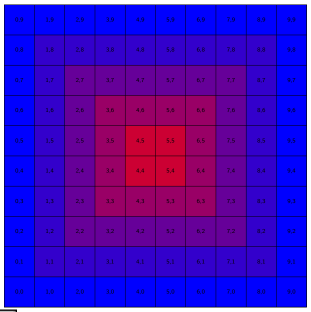
- svg - Scalable vector graphics format
- HEIC - high efficiency image format
- TIFF - tagged image file format. high resolution files with poor compression, popular with scanners and microscopy equipment. Legacy support is sometimes available.

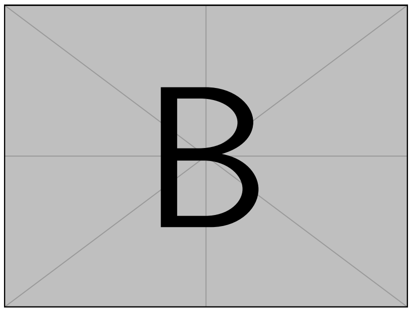
The mwe (minimum working example) package provides backend support for sample images. These are:

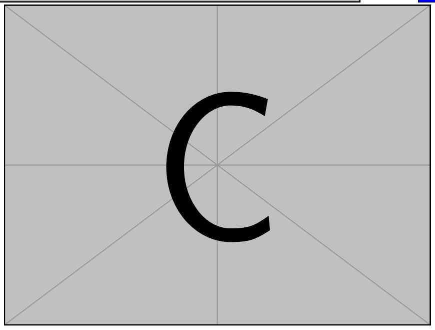
- example-image-a
- example-image-golden
- example-grid-100x100pt
- example-image-b
- example-image-c
- example-image

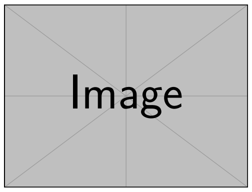












There are a few common options you will need:

size A scale factor (a float) for the image size. Default is 1

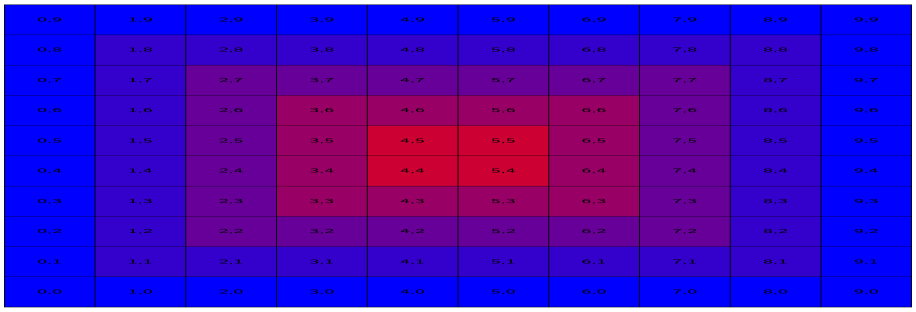
angle Rotation, 360 degree scale. Default is 0

width equals a unit (typically #cm), sets image scale by width

height equals a unit (typically #cm), sets image scale by width

If both height and width are set the aspect ratio is broken. Use size to avoid this. Example

```
\includegraphics[width=12cm, height=4cm]{example-grid-100x100pt}
```



Useful for tuning plot sizes to fit into a specific spot with vector images (especially on posters).

8 Figures, tables, and tabular

8.1 Figures

Figures are created with a figure environment. Figures are an example of a floating environment, LaTeX places restrictions on where figures can go (figures can't be split across pages and there are 19 others you don't need to know). For more details see this stackexchange post.

- Within the environment, add \centering to center the figure contents.
- To change the position of a figure, pass an optional argument to the figure environment. There are 5 optional arguments:
  - h for inline placement
  - ! for relax placement rules
  - b for bottom of page
  - t for top of page
  - p for place in a special place for floats (either a column or page at the end of the document
- There is *no* precedence in the order of float insertion. [!hpt] has the same effect as [pht!].
- Floats with a ! are sometimes called bang floats (as the #! is called a she-bang or a hashbang, Unix is awesome like that).
- Figures can contain a caption, which will add the figure number. All figures should have a caption. To add a short caption (eg figure name), enter it as an optional argument to caption.

```
\begin{figure}[h!]
\centering
\includegraphics[width=3cm]{example-grid-100x100pt}
\caption[Grid demo fig]{This is a caption for the sample figure}
\end{figure}
```

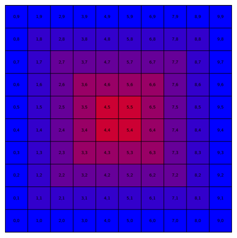


Figure 1: This is a caption for the sample figure

8.2 Tables

Tables behave like figures, but follow formatting convention rules for tables with regard to captions. The actual table itself is generated with a tabular environment.

- The tabular environment accepts a string that describes how to build columns. There are 4 common characters you can use:
  - | (pipe) adds a vertical line
  - l left justify the column
  - c center justify
  - r right justify

- To advance to the next cell, use an &. To advance to the next line, use a \\. You are responsible for tracking the number of cells in your table and your current position in the table.
- Horizontal lines can be added with \hline. Two hline calls produce a break in | constructed lines.

A simple table:

```
\begin{table}[h!]
\centering
\begin{tabular}{c c}
a & $x^2$ \\
c & 23
\end{tabular}
\caption[Sample table]{This is an example of a table. We can put inline math in tabular cells.
```

a	x^2
c	23

Table 1: This is an example of a table. We can put inline math in tabular cells.

We can make a slightly more complicated table to demonstrate these features

n	x_n	$f(x_n)$	ϵ_n
1	12.2	12.21	0.01
2	12.3	12.32	0.12
3	12.4	12.43	0.23

Table 2: This is a second example of a table. .

```
\begin{table}[h!]
\centering
\begin{tabular}{|l|c c r|} \hline
n & $x_n$ & $f(x_n)$ & $\epsilon_n$ \\  \hline \hline
1 & $12.2$ & $12.21$ & $0.01$ \\
2 & $12.3$ & $12.32$ & $0.12$ \\
3 & $12.4$ & $12.43$ & $0.23$ \\
\hline
\end{tabular}
\caption[Advanced table]{This is a second example of a table. .}
\end{table}
```

9 References

LaTeXsupports 2 types of references: citations and labels. To reference an equation/figure/table/theorem/section/*environment, it needs to have a label. Labels are created with \label{label-key}. A recommended naming scheme is a 3 letter key for each type of object:

- eqn for equation
- thm for theorem
- fig for figure
- tbl for table
- lem for lemma
- rmk for remark
- cor for corollary
- prt for part
- cha for chapter
- sec for section
- sbs for subsection
- ssb for subsubsection

followed by a terse description of the object, separated by a colon. Many IDEs support drop-down text-completion for labels, but these are typically limited to $<$ 25 characters. Examples: thm:completeness-of-R, eqn:integral-minimizer, fig:x causes y to change. Spaces are allowed, the choice between spaces, dashes, and underscores is personal preference. Dashes are usually pretty easy to see, whereas spaces and underscores can be combined (is it 1 or 2 spaces/underscores?), especially if you're not using a monospace font in your editor.

- Figure references should follow the caption.
- Table references should follow the caption.
- Equation references (or align or gather) can go anywhere in the environment.
- Theorem environment references can go anywhere in the environment.

The caption call generates the label-able object in floats. Access a labeled object with \ref{}. If no object is found, a **??** is returned.

Additionally, a table of contents can be printed with \tableofcontents.

Contents

A list of figures can be generated with \listoffigures.

List of Figures

A list of tables can be generated with \listoftables.

List of Tables

Only one call of each can be made per document.

10 Abstract

Abstracts can be included with the abstract environment. This environment is part of the base latex distribution and requires no packages.

```
\begin{abstract}
Place your abstract here.
\end{abstract}
```

11 Including documents

There are 3 ways to include documents. All three of these methods are comparable with pdflatex.

1. \include{path/to/file.tex} Dumps in the contents of file.tex at compile time. You need to compile the main document and not file.tex. \include can be nested (file.tex can also have a \include call).
2. \input{path/to/file} behaves similar to \include, but inserts a page break. Good for books with chapters.
3. Use the pdfpages package. This allows you to insert pdf documents into your compiled document. This is particularly useful for joining PDF files from multiple programs. For example, Mathematica and Matlab have robust print methods to produce documents showing your code and the outputs. Other notebook style code editors (such as R markdown or Jupyter notebooks) produce PDF files you might want to write in their native languages, rather than attempt to recreate the cells in LaTeX.

12 Including code snippets

For most popular languages, the listings and minted packages provides code import.

- listings is a highly customizable tool that can be finely tuned.
- Minted is good with default parameters. Minted is built on top of a python package called Pygments (like 'pigments'), the minted package contains an API to interface with Pygments. You can install pygments easily with pip (the python package manager) with cmd if you have a python installation. Use "pip install Pygments".
- If you are using a standalone install of LaTeX, you will need to adjust your editor pdflatex call to include -shell-escape. My build command (in TeXStudio) looks like:

  ```
  -synctex=1 -interaction=nonstopmode -shell-escape %.tex
  ```
- Overleaf supports minted with no need to install anything.

12.1 Python

With listings only specifying python as the language.

```
c = 0
for x in range(12):
        c = c + x
print(c,'hello␣world')
```

With minted only specifying python as the language.

```
c = 0
for x in range(12):
        c = c + x
print(c,'hello world')
```

```
\lstinputlisting[language=python]{code/python.py}
\inputminted{python}{code/python.py}
```

12.2 C

With listings only specifying C as the language.

```
#include <stdio.h>
int main() {
   // printf() displays the string inside quotation
   printf("Hello,␣World!");
   return 0;
}
```

With minted only specifying C as the language.

```
#include <stdio.h>
int main() {
   // printf() displays the string inside quotation
   printf("Hello, World!");
   return 0;
}
```

```
\lstinputlisting[language=C]{code/cdemo.c}
\inputminted{C}{code/cdemo.c}
```

12.3 Matlab

There is a specific package matlab-prettifier built on top of listings for matlab code. We can specify the style as matlab-editor with an optional argument.

```
x = 1;
for y = 1:n
        x = x + 1
end
disp(x)
```

with Minted:

```
x = 1;
for y = 1:n
        x = x + 1
end
disp(x)
```

```
\lstinputlisting[style=matlab-editor]{code/Matlab.m}
\inputminted{Matlab}{code/Matlab.m}
```

Listings is not currently under developement while minted has an active community. Installation aside, Minted is easier to use and has many more advanced features (we could spend an hour talking just about minted). You're encouraged to read the documentation for minted for more details. Minted supports captions, references/labels, can float code snippets, inline snippets, mixing hilighting (you can compare between multiple languages) and tables of contents for code floats. It's a very cool piece of software.

Line numbering and boxing is supported by both listings and minted.

```
\definecolor{bg}{rgb}{0.95,0.95,0.95} % define a nice color gray
\begin{listing}[H]
\inputminted[linenos=true, bgcolor=bg]{C}{code/cdemo.c}
\caption{A simple C program.}
\label{lst:c-demo}
\end{listing}
Listing \ref{lst:c-demo} contains a simple C program.

We can generate a list of all listings with minted:
\listoflistings
```

```c
#include <stdio.h>
int main() {
   // printf() displays the string inside quotation
   printf("Hello, World!");
   return 0;
}
```

Listing 1: A simple C program.

Listing 1 contains a simple C program.

We can generate a list of all listings with minted:

List of Listings

We can even put LaTeX markdown code in a file and get it to compile!

```
\begin{listing}[H]
\begin{minted}[linenos=true, bgcolor=bg, mathescape]{python}
# computes sum $\sum_1^k \frac{1}{k}$
k = 12
total = 0
for x in range(k):  # Loop over the first k integers
```

```
total += 1/x  # Increment our total
print(total)
\end{minted}
\caption[Minted is Awesome]{Our python file has \LaTeX{} in the comments!}
\end{listing}
```

```python
# computes sum $\sum_1^k \frac{1}{k}$
k = 12
total = 0
for x in range(k):  # Loop over the first k integers
    total += 1/x  # Increment our total
print(total)
```

Listing 2: Our python file has LaTeX in the comments!

13 Bibliographies

We can create citations with \cite{}. More advanced tools are available. In LaTeX, a .bib file format denotes bibliographic information. A simple one looks like this:

```
@article{Finch1963,
author = {Finch, P D},
doi = {10.1017/S1446788700028378},
issn = {14468107},
journal = {Journal of the Australian Mathematical Society},
number = {3},
pages = {351--358},
title = {{A limit theorem for Markov chains with continuous state space}},
url = {https://doi.org/10.1017/S1446788700028378},
volume = {3},
year = {1963}
}
```

The first line is called the citation key. We can attach a bibliography file with the \bibliography{file-name} command. The cite command can accept multiple keys (comma separated) such as[Fin63, Lud75]. We can make a reference to this citation with \cite{Finch1963} which produces [Fin63]. A good IDE will provide you a list of citation keys in the declared bibliography.

- The first line is called the citation key.
- We can attach a bibliography file with the \bibliography{file-name} command.
- We can make a reference to this citation with \cite{Finch1963} which produces [Fin63].
- A good IDE will provide you a list of citation keys in the declared bibliography.
- We need to declare a bibliography style (\bibliographystyle{}) that provides pdflatex with the formatting information to use. Options include plain, acm, alpha, acm, and siam. This example uses alpha.

- if you change the style and it doesn't update, delete the extra files generated by bibtex and the aux file. This will provide a clean slate for the compiler's build sequence.

13.1 Bibtex

This bibliography is generated with

```
\bibliographystyle{alpha}
\bibliography{demo_bib}  % Omit file suffix
```

References

[Fin63] P D Finch. A limit theorem for Markov chains with continuous state space. *Journal of the Australian Mathematical Society*, 3(3):351–358, 1963.

[Lud75] Donald Ludwig. Persistence of Dynamical Systems under Random Perturbations. *SIAM Review*, 17(4):605–640, 1975.

For most classes this will be enough. This uses a process called bibtex (bibtex is the executed file, bibber is the backend). Bibtex is simple to use, but lacks advanced features. Natbib is a depreciated alternative, but Biblatex is replacing both of them.

13.2 Biblatex

Biblatex is a more advanced package.

14 SI Unit formatting

The package SIunitX is a great package designed for formatting scientific calculations (chemistry, physics, etc). The general formatting is \SI {value (in 1e-3 style scientific notation)}{Units}. To declare a unit, use \unit-name. for a per-second unit structure, add \per \unit. We can get values like: $1 \times 10^{-4}\,\mathrm{m\,s^{-1}}$. Read the docs here.

15 External referencing

We've talked about internal referencing, but what about external referencing? Wouldn't it be great if you could embed an email address (or a click-and-mail style event) or a website link in your document? The hyperref package is your answer. There are 2 quick tools:

- href provides an embeded style reference of the formula \href{embedded link or action}{displayed text}
  1. To embed an email use mailto:user@server.web, the mailto: prefix determines the action
  2. To embed a link, just paste in the link.
- \url{} is a lightweight version of href that does not hide your link. Great if you want to link to `google.com`. Less good if you want `https://en.wikipedia.org/wiki/Hyperlink#/media/File:Wiki-linking.png`

Read the hyperref documentation here and the overleaf tutorial here.