LaTeX for beginners using CoCalc

Valerio De Angelis

Spring 2025

Preface

This is a beginner's guide to preparing documents in LaTeX using the web based system CoCalc (www.cocalc.com). It assumes no prior knowledge of LaTeX, CoCalc, or computer programming.
This document is a modified version of the document "LaTeX for beginners", prepared by the Information Services Office of the University of Edinburgh, https://www.ed.ac.uk/information-services Edition 5, March 2014 Document Reference: 3722-2014, Copyright©IS 2014.
This modification of the original workbook is designed to be used with the Math 4005 course 'Advanced and Experimental Problem Solving' at Xavier University of Louisiana.

Contents

1 Introduction

1.1 What is LaTeX?

LaTeX is an open source software widely used by mathematics researchers, instructors and students to prepare professional-looking documents. [1]

Proficiency in using LaTeX has become an essential part of most mathematics programs in the US and elsewhere, both graduate and undergraduate. Virtually all mathematics theses or dissertations are written in LaTeX. Students who graduate in Mathematics with little or no knowledge of LaTeX are sure to be at a disadvantage if they wish to pursue further graduate or professional studies.

The LaTeX software is built on TeX, a digital typesetting system originally conceived and written by Donald Knuth in the 1980's when he became disappointed by the quality of mathematical typesetting available at the time. As such, TeX and its descendant LaTeX are especially good at typesetting mathematical expressions of almost any complexity, because they were originally designed exactly for that purpose.

But from a user's point of view, the main difference between LaTeX and a word processor (such as Microsoft Word) is not its ability to handle complex mathematical typesetting. The difference is that LaTeX is specifically conceived to make users focus on the *content* of what they are writing, and not on how it will look on the paper. By contrast, a word processor such as Microsoft Word is of the WYSIWYG type (What You See Is What You Get): what you see on the screen as you type is exactly what will be printed on your final document. This means that the program will not provide any help in structuring and formatting the document appropriately. In this sense, a better description of a WYSIWYG system is What You See Is All You Get.

With LaTeX you concentrate on the logical structure of what you are writing, not on how it will look. A typical example is the use of **boldface** text. Most of the time people use it in order to emphasize a word. But there are other ways to do that: we could use *italic* or ALL CAPS with a similar effect. When writing in LaTeX we do not worry about how to best emphasize a word. We type instead `\emph{word}` where `word` is the word to be emphasized, and then let LaTeX decide how to render that in print. Similarly, if we want to divide our document in different units such as chapters, sections, or subsections, we do not worry how and where we should place the title of each unit, what font to use, how to number them, etc. We simply write `\chapter{chaptertitle} \subsection{sectiontitle}`, and so on, and LaTeX will do all the right formatting for us.

To prepare a LaTeX document, we need to write the content in a plain text file with extension .tex and with some appropriate code at the beginning. Then processing the .tex file with LaTeX will produce the output as a pdf document that can be saved, mailed or printed. While this process requires some significant preparation using a standard LaTeX package for personal computers, it is made much easier and accessible by the web based Co-Calc system, that we will use in this course. No special software needs to be downloaded. All you need

[1] In spite of the X at end of its name, it is pronounced *lay-tek*. That's because the letters T e X stand for the Greek letters τ (tau), ϵ (epsilon), χ (chi) and derive from the Greek $\tau\epsilon\chi\nu\eta$, that is the origin of the English word *technology*. So that is the word you should think about when pronouncing the TeX in LaTeX.

is a computer connected to the internet and a web browser.

1.2 Before You Start

The following conventions are used throughout this workbook:

- Actions for you to carry out are bulleted with an arrow ⇒.
- `Text you type is written in this font.`
- Menu commands, button names and other text appearing on the CoCalc page are shown in **bold**.

Although the code in this workbook should work in any LaTeX editor, specific examples and screenshots refer to the web based cloud computing and course management platform for computational mathematics CoCalc (https://cocalc.com) as available in January 2022.

2 Document Structure

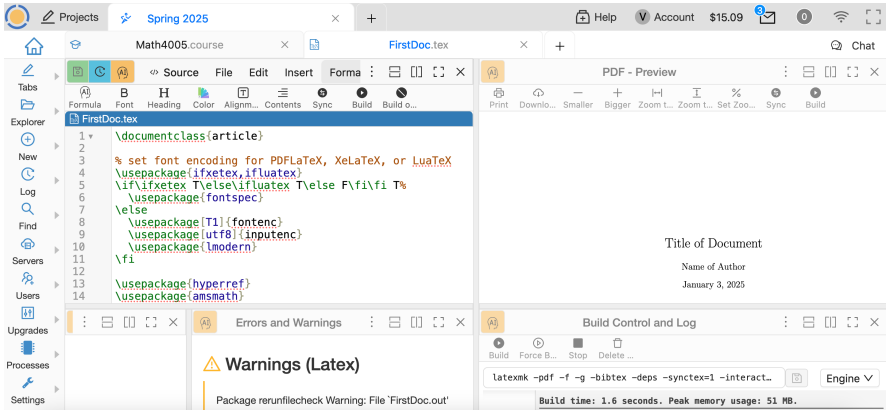
2.1 Essentials

⇒ Login to your CoCalc account (or create one).

⇒ Open the project you wish to work on (or create a new one), and navigate to (or create) the folder where you want to store the LaTeX document. To create new folder, click on the arrow next to **New** then select **folder** at the bottom of the menu.

⇒ Click on the big blue box **Create or Upload Files...**, then enter a name for your document in the text box, then click on **LaTeX document**. In the picture below we are using the filename *FirstDoc*.

After a short wait, your screen will look like this:



Note that there are four different frames, two on the left and two on the right.

- The top left side is the 'LaTeX source code' frame, and that is where you will type the content of your document.
- The top right side shows the finished product, as a pdf file. This is the 'PDF - Preview' frame.
- The bottom left is a 'Errors and Warnings' frame, used for troubleshooting.
- The bottom right is a 'Build Control and Log' frame.

By clicking anywhere on a frame, you will notice that several icons appear on the top bar. On all frames except the 'PDF - Preview' frame, you will find the 'Build' icon.

⇒ Place your cursor on the various icons on the top left frame and notice the comment that appear.

⇒ Now click on the top right window and notice the icons that appear on the top bar. Explore the comments for the icons.

Note that a standard generic template is automatically loaded in the 'LaTeX source code' frame, and it contains line numbers. The line numbers will make it easier to compare your code with the screenshots and find errors.

2.2 The preamble and title

We will now analyze the content of the automatically loaded LaTeX code on the top left frame line by line.

- Line 1: **\documentclass{article}**

  This command must appear at the start of every LaTeX document, and it specifies the type. Besides 'article' (that we will use in this workbook), other available types are 'report', 'book', 'letter', 'slides', 'standalone', etc. Note the backslash \ before the command **documentclass**. All LaTeX commands are preceded by a backslash. The curly braces {} are used to enter mandatory input. If you leave them out, you will get an error message. In this case, the mandatory input is the type of document we are preparing. If a command also accepts optional input, the square brackets [ ] are used. So for example
  \documentclass[twocolumn]{article} is used to format the document on two columns instead of one. Anything typed after
  \documentclass{article} and before **\begin{document}** is known as the *preamble*, and will affect the whole document.

- Notice several lines that contain the code `\usepackage`. This is used to load blocks of code (called *packages*) that enhance the functionality of LaTeX . We will soon make use of a package. Now jump to the line containing **\title** (line 16 in the CoCalc version as of this writing).

- **\title{Title of Document}** This is the title of the document, as you can see from the pdf output on the right.

⇒ As your first interaction with the system, change **Title of Document** to `Math 4005`, and watch as the system updates the pdf output. You may have to wait several seconds, up to a minute or so. If you do not want to wait, click the **Build** icon on the top bar.

- **\author{Name of Author}** This is the author.

⇒ Change **Name of Author** to your name, and click on **Build** to see the update.

- Note that there is no code for the date, but the pdf output displays today's date by default.

⇒ Add the code `\date{January 14, 2025}` to force the date to be a different one, and click **Build**.

- If we prefer not to have any date displayed, we use `\date{}`

⇒ Change `\date{January 14, 2025}` to `\date{}` and click on **Build**.

- Ignore the next few lines, up to the line containing **\begin{document}**. This is the command that tells LaTeX where the content of the document begins.

- The next line is **\maketitle**. This is telling LaTeX to actually display the title, author and date.

- In the CoCalc template, there is no other input until the **\end{document}** line, that tells LaTeX where the document content ends. All input after the **\end{document}** line will be ignored.

⇒ Enter the text `This is my first document` anywhere after **\begin{document}** and before **\end{document}**, and click **Build**.

- Typing a short sentence as you did in the previous item is not going to illustrate what a full page of text will look like in LaTeX . We will now make use of a package whose only aim is to fill a full page with already prepared text.

⇒ Go back to the second line (just after `\documentclass{article}`) and type `\usepackage{lipsum}`

⇒ Go the line where you typed `This is my first document` and replace it with `\lipsum`

⇒ Click on build. You will see over a page of some Latin (and meaningless) text. The `\lipsum` command can also be used with an optional argument. So `\lipsum[1]` will print only the first paragraph of the same Latin text, and `\lipsum[1-3]` will print the first three paragraphs.

⇒ We now experiment with optional input for a command. Replace **\documentclass{article}** on Line 1 with `\documentclass[twocolumn]{article}` and click on **Build** to see the result.

Notice that there are several lines in the source code shown in red, and with a % sign at the beginning. Every line that begins with the % sign is ignored by LaTeX . This feature is used to add comments to the source code, that will help you remember why you used some code and what it does. It can also be used (as it is here) to exclude from the current document some piece of code that we may want to use on another document using the same template. Comments will be discussed again in Section 3.4.

You will also notice several empty lines in the source code frame. The empty lines aren't necessary[2], but they will make it easier to navigate between the different parts of the document as it gets longer.

2.3 Troubleshooting

If there are some errors in the code, error messages will appear in the bottom left frame. These messages often provide useful information to find what the problem is. Experiment by removing the backslash from
`\documentclass{article}` and click on **Build**. You will see error messages in the bottom left frame, and no output in the PDF - Preview frame. Reading the error messages in the bottom left frame we can easily find out that the **documentclass** command is missing.

Sometimes the system will try to display some output even when errors are present. Experiment by adding a % sign at the beginning of the line with `\usepackage{lipsum}`. The PDF - Preview frame should now show some partial content, but there is an error message because LaTeX cannot find the command `\lipsum`.

2.4 Sections

You should divide your document into chapters (if needed), sections and sub-sections. The following sectioning commands are available for the article class:

- `\section{...}`
- `\subsection{...}`
- `\subsubsection{...}`
- `\paragraph{...}`
- `\subparagraph{...}`

The title of the section replaces the dots between the curly brackets. With the **report** and **book** classes we also have `\chapter{...}`.

• ⇒ Remove the optional argument `[twocolumn]` from the first line and the % sign from the `\usepackage{lipsum}` line, and replace `\lipsum` with the following:

```
\section{Introduction}
\lipsum[1]
```

[2]See section 3.4 on page 8 for information about how LaTeX deals with empty space in the .tex file.

```
\section{The problem I am solving}
\lipsum[2]
\subsection{First step}
\lipsum[3]

\subsection{Second step}
\lipsum[4]

\section{Answer}
\lipsum[5]
```

⇒ Click on **Build**. You will see how your document is now organized in numbered sections and subsections.

2.5 Labelling

You can label any of the sectioning commands so they can be referred to in other parts of the document. Label the section with `\label{labelname}`. Then type `\ref{labelname}` or `\pageref{labelname}`, when you want to refer to the section or page number of the label.

⇒ Type `\label{probl}` on a new line directly below `\section{The problem I am solving}`.

⇒ Type `Referring to section \ref{probl} on page \pageref{sec1}` at the beginning of the **Answer** section, and check the PDF-Preview frame.

You will see in the pdf output a clickable box with red border and with the number of the section inside, that when clicked will take you to Section 2.

2.6 Table of Contents

If you use sectioning commands it is easy to generate a table of contents. Type `\tableofcontents` where you want the table of contents to appear in your document — often directly after the title page.

You may also want to change the page numbering so that roman numerals (i, ii, iii) are used for pages before the main document starts. This will also ensure that the main document starts on page 1. Page numbering can be switched between arabic and roman using `\pagenumbering{...}`.

⇒ Type the following on a new line below `\maketitle`:

```
\pagenumbering{roman}
\tableofcontents
```

```
\newpage
\pagenumbering{arabic}
```

The `\newpage` command inserts a page break so that we can see the effect of the page numbering commands. Click on **Build** and check the PDF-Preview frame.

3 Typesetting Text

3.1 Font Effects

There are LaTeX commands for a variety of font effects:

`\textit{words in italics}`	*words in italics*
`\textsl{words slanted}`	*words slanted*
`\textsc{words in smallcaps}`	WORDS IN SMALLCAPS
`\textbf{words in bold}`	**words in bold**
`\texttt{words in teletype}`	`words in teletype`
`\textsf{sans serif words}`	sans serif words
`\textrm{roman words}`	roman words
`\underline{underlined words}`	underlined words

⇒ Add some more text to your document and experiment with different text effects.

3.2 Font Sizes

There are LaTeX commands for a range of font sizes:

`{\tiny tiny words}`	tiny words
`{\scriptsize scriptsize words}`	scriptsize words
`{\footnotesize footnotesize words}`	footnotesize words
`{\small small words}`	small words
`{\normalsize normalsize words}`	normalsize words
`{\large large words}`	large words
`{\Large Large words}`	Large words
`{\LARGE LARGE words}`	LARGE words
`{\huge huge words}`	huge words

⇒ Experiment with different font sizes in your document.

3.3 Lists

LaTeX supports two types of lists: **enumerate** produces numbered lists, while **itemize** is for bulleted lists. Each list item is defined by `\item`. Lists can be nested to produce sub-lists.

⇒ Type the following to produce a numbered list with a bulleted sub-list:

```
\begin{enumerate}
\item First thing
\item Second thing
\begin{itemize}
\item A sub-thing
\item Another sub-thing
\end{itemize}
\item Third thing
\end{enumerate}
```

⇒ Click on the **Build** button and check the PDF.

The list should look like this:

1. First thing
2. Second thing
   - A sub-thing
   - Another sub-thing
3. Third thing

It is easy to change the bullet symbol using square brackets after the `\item`, for example, `\item[-]` will give a dash as the bullet. You can even use words as bullets, for example, `\item[One]`.

Code:

```
\begin{itemize}
\item[-] First thing
\item[+] Second thing
\begin{itemize}
\item[Fish] A sub-thing
\item[Plants] Another sub-thing
\end{itemize}
\item[Q] Third thing
\end{itemize}
```

Output:

- First thing

+ Second thing

  Fish A sub-thing

  Plants Another sub-thing

Q Third thing

3.4 Comments and Spacing

Comments are created using %. When LaTeX encounters a % character while processing a .tex file, it ignores the rest of the line (until the [**Return**] key has been pressed to start a new line — not to be confused with line wrapping in your editor). This can be used to write notes in the input file which will not show up in the printed version. In the following code:

```
The French revolution of 1789 had a profound effect on the
```

```
European balance of power, % remember to add a reference here

and it took place 13 years after the American

declaration of independence.
```

the text after the % is a comment for the writer, not to be shown in the output, that will be:

The French revolution of 1789 had a profound effect on the European balance of power, and it took place 13 years after the American declaration of independence.

Multiple consecutive spaces in LaTeX are treated as a single space. Several empty lines are treated as one empty line. The main function of an empty line in LaTeX is to start a new paragraph. In general, LaTeX ignores blank lines and other empty space in the .tex file. Two backslashes (\\) can be used to start a new line.

⇒ Experiment with putting comments and blank lines in your document.

If you want to add vertical blank space into your document use the `\vspace{...}` command. This will add blank vertical space of a height specified in typographical points (pt) or other units. For example, `\vspace{12pt}` will add space equivalent to the height of a 12pt font, and `\vspace{0.5in}` will add a vertical space of 0.5 inches. Other units that can be used are ex, em (approximately the space taken by the letters x or m), and cm. Note that `\vspace` needs to be used after a blank line, or else it will have no effect.

In a similar way, `\hspace{...}` is used to insert horizontal blank space. To start a new page, use the command `\newpage`.

3.5 Special Characters

The following symbols are reserved characters which have a special meaning in LaTeX:

```
#   $   %   ^   &   _   {   }   ~   \
```

All of these apart from the backslash \ can be inserted as characters in your document by adding a prefix backslash:

```
\#   \$   \%   \^{}   \&   \_   \{   \}   \~{}
```

The above code will produce:

$ % ^ & _ { } ~

Note that you need to type a pair of curly brackets `{}` after the hat `^` and tilde `~`, otherwise these will appear as accents over the following character. For example, `\^e` produces ê.

The backslash character \ can not be entered by adding a prefix backslash, \\, as this is used for line breaking. Use the `\textbackslash` command instead.

⇒ Type code to produce the following sentence in your document:

Item #1A\642 costs $8 & is sold at a ~10% profit.

Send email to your instructor or check the .tex file of this workbook if you need help.

3.6 Colored Text

To put colored text in your document you need to use a **package**. There are many packages that can be used with LaTeX to enhance its functionality. Packages are included in the **preamble** (i.e. before the `\begin{document}` command). Packages are activated using the `\usepackage[options]{package}` command, where `package` is the name of the package and `options` is an optional list of keywords that trigger special features in the package.

The basic color names that `\usepackage{color}` knows about are black, red, green, blue, cyan, magenta, yellow and white:

Red, green, blue, cyan, magenta, yellow and white.

The following code produces colored text:

```
{\color{color_name}text}
```

Where `color_name` is the name of the color you want, and `text` is the text you want to be colored.

⇒ Type `\usepackage{color}` on the line before `\begin{document}`.

⇒ Type `{\color{red}fire}` in your document.

⇒ Click on the **Build** button and check the PDF.

The word 'fire' should appear in red.

It is possible to add options that allow `\usepackage{color}` to understand more color names, and even to define your own colors. It is also possible to change the background color of text. To obtain yellow use the command `\colorbox{black}{\color{yellow}yellow}`. If you want more information see the Colors chapter in the LaTeX Wikibook[3].

4 Equations

One of the main reasons for writing documents in LaTeX is because it is really good at typesetting equations. Equations are written in *math mode*.

[3] `http://en.wikibooks.org/wiki/LaTeX/Colors`

4.1 Inserting Equations

You can enter math mode with an opening and closing dollar sign `$`. This can be used to write mathematical symbols within a sentence — for example, typing `$1+2=3$` produces $1 + 2 = 3$. Math expression written this way are called *inline*.

If instead of an inline math expression you want a *displayed* equation on its own line use `$$...$$`.

For example, `$$1+2=3$$` produces:

$$1 + 2 = 3.$$

There are many math expressions that can only be typed using math mode (as we will soon see with exponents, subscripts, etc.). But even when an expression is simple enough that no special commands are necessary (such as `5x-3x=2x`), it is important to use math mode every time we are typing math symbols. Compare the outputs resulting from typing `5x-3x=2x` with and without math mode:

Source code	Output
`5x-3x=2x`	5x-3x=2x
`$5x-3x=2x$`	$5x - 3x = 2x$

As you can see, the minus sign − used in math typesetting is quite different from the hyphen - produced by the computer keyboard, the spacing between symbols is different, and the font of mathematical variables such as x is rendered in italic. Any reader with even moderate experience in mathematical typesetting will recognize an equation such as 5x-3x=2x as incorrectly typed in LaTeX without using math mode.

For a numbered displayed equation, use `\begin{equation}...\end{equation}`.

For example, `\begin{equation}1+2=3\end{equation}` produces:

$$1 + 2 = 3 \tag{1}$$

The equation label (4.1) that you see at the right refers to Chapter 4, Section 1. This will only appear if you are using a document class with chapters, such as `report`.

Use `\begin{eqnarray}...\end{eqnarray}` to write equation arrays for a series of equations or inequalities. For example —

```
\begin{eqnarray}
  a & = & b + c \\
    & = & y - z
\end{eqnarray}
```

Produces:

$$\begin{aligned} a &= b + c && (2)\\ &= y - z && (3) \end{aligned}$$

For unnumbered equations add the star symbol `*` after the `equation` or `eqnarray` command (i.e. use `{equation*}` or `{eqnarray*}`).

4.2 Mathematical Symbols

Although some basic mathematical symbols such as + - * / = can be accessed directly from the keyboard, most must be inserted using a command.

This section is a very brief introduction to using LaTeX to produce mathematical symbols — the Mathematics chapter in the LaTeX Wikibook is an excellent tutorial on mathematical symbol commands, which you should refer to if you want to learn more. If you want to find the command for a specific symbol try Detexify[4], which can recognize hand drawn symbols.

4.2.1 Powers and Indices

Exponents are inserted using the hat `^` symbol. For example, `$n^2$` produces n^2.

Subscripts are inserted using an underscore `_`. For example, `$x_2$` produces x_2.

If the exponent or subscript includes more than one character, group them using curly brackets `{...}`, for example `e^{x^2}` produces e^{x^2} and `$x_{n-2}$` produces x_{n-2}.

4.2.2 Fractions

Fractions are inserted using `\frac{numerator}{denominator}`.

`$$\frac{a}{3}$$` produces:

$$\frac{a}{3}$$

Fractions can be nested.

`$$\frac{y}{\frac{3}{x}+b}$$` produces:

$$\frac{y}{\frac{3}{x}+b}$$

4.2.3 Roots

Square root symbols are inserted using `\sqrt{...}` where `...` is replaced by the square root content. For cube roots or higher roots, use the optional square brackets `[...]`.

`$$\sqrt{2y}$$` produces:

$$\sqrt{2y}$$

`$$\sqrt[3]{2y}$$` produces:

$$\sqrt[3]{2y}$$

[4] `http://detexify.kirelabs.org`

4.2.4 Inequalities, Other symbols

The following table shows the code to produce the symbols we are likely to use in this course.

Source code	Output	Source code	Output
`$\leq$`	$\leq$	`$\cap$`	$\cap$
`$\geq$`	$\geq$	`$\subset$`	$\subset$
`$\neq$`	$\neq$	`$\emptyset$`	$\emptyset$
`$\equiv$`	$\equiv$	`$\exists$`	$\exists$
`$\in $`	$\in$	`$\forall$`	$\forall$
`$\notin$`	$\notin$	`$\rightarrow$`	$\rightarrow$
`$\infty$`	∞	`$\Rightarrow$`	$\Rightarrow$
`$\cup$`	$\cup$	`$\Leftrightarrow$`	$\Leftrightarrow$

4.2.5 Limits, Sums and Integrals

The command `$$\lim_{x\rightarrow \infty}f(x)$$` produces

$$\lim_{x\rightarrow\infty} f(x).$$

The command `\sum` inserts a sum symbol; `\int` inserts an integral. For both symbols, the lower limit is specified by an underscore character _, and the upper hat limit by a hat character ^.

`$$\sum_{k=1}^5 x^k$$` produces:

$$\sum_{k=1}^{5} x^k$$

`$$\int_a^\infty f(x)dx$$` produces:

$$\int_a^\infty f(x)dx.$$

The `\lim`, `\sum` and `\int` commands are displayed differently in inline mode. So `$\lim_{x\rightarrow \infty} f(x)$` produces $\lim_{x\rightarrow\infty} f(x)$, `$\sum_{k=1}^5 x^k$` produces $\sum_{k=1}^{5} x^k$ and `$\int_a^b f(x)dx$` produces $\int_a^b f(x)dx$. Sometimes we want to use the displayed form for a limit, sum or integral without actually having to display it. In this case we can use the command `\displaystyle`. So `$\displaystyle \lim_{x\rightarrow \infty} f(x)` produces $\displaystyle \lim_{x\rightarrow\infty} f(x)$,

`$\displaystyle \sum_{k=1}^5 x^k$` produces $\displaystyle \sum_{k=1}^{5} x^k$ and

`$\displaystyle \int_a^b f(x)dx$` produces $\displaystyle \int_a^b f(x)dx.$

In a similar way, the `$\frac{1}{2}$` command used inline results is a smaller fraction: $\frac{1}{2}$. If we prefer to have the full size fraction for inline output, we can use `$\dfrac{1}{2}$`: $\dfrac{1}{2}$.

4.2.6 Greek letters

Greek letters can be typed in math mode using the name of the letter preceded by a backslash \. Many Greek capital letters are the same in the Latin alphabet. For example, capital α (the Greek letter alpha) is A, the same as capital a. For those that are different capitalize the first letter of the name to produce a capital Greek letter.

For example:

`$\alpha$` $= \alpha$
`$\beta$` $= \beta$
`$\delta, \Delta$` $= \delta, \Delta$
`$\theta, \Theta$` $= \theta, \Theta$
`$\mu$` $= \mu$
`$\pi, \Pi$` $= \pi, \Pi$
`$\sigma, \Sigma$` $= \sigma, \Sigma$
`$\phi, \Phi$` $= \phi, \Phi$
`$\psi, \Psi$` $= \psi, \Psi$
`$\omega, \Omega$` $= \omega, \Omega$

4.2.7 Mathematical functions and text in math mode

In properly typeset mathematics, the common mathematical functions exp, log, ln, all the trig functions $\sin, \cos, \tan, \sec, \csc, \cot$ and the inverses arcsin, arccos, arctan as well as max, min, sup, inf, should not be italicized.

But if we type them in math mode (as it is natural to do, because they will occur in math expressions), LaTeX will interpret them as variables multiplied together instead of names of math functions, and so it will render them in italic.

So typing `$cos(2x)$` will produce $cos(2x)$ instead of the correct $\cos(2x)$. LaTeX thinks that cos means the variable c times the variable o times the variable s. In order to avoid this, we need to place a backslash in front of each of these functions. So to produce $\cos(2x)$, use `$\cos(2x)$`, to produce $\ln(1-x)$ use `$\ln(1-x)$`, and so on.

A similar problem occurs if we need to write a few plain text words inside an expression in math mode. Suppose for example we want to produce the output

$$\cos(n\pi) = -1 \text{ if } n \text{ is odd } .$$

If we use the code `$$\cos(n\pi) =-1  if  n  is odd.$$` we will get the output

$$\cos(n\pi) = -1 if n is odd.$$

This is quite bad, not just because the text is in italic, but also because LaTeX ignores spaces when it thinks the letters are all variables multiplied together. To avoid this, we place any text inside a math expression in the argument of the command `\mbox{...}`. So the correct code for the above example is `$$ \cos(n\pi) =-1  \mbox{ if } n \mbox{ is odd } .$$` Note that we added a space before and after the words. Otherwise LaTeX will write them with the right font, but without the proper spacing.

4.3 Delimiters

A delimiter is any symbol that is used (normally in pairs) to group together symbols or expressions. The most common delimiters are the parentheses: ( ), the brackets [ ], and the curly brackets { }. Remember from the previous section that the curly brackets are a special character for LaTeX and we need to type `\{` or `\}` to obtain them.

Other delimiters are the vertical bars | | , the floor function ⌊ ⌋ (produced with the `\lfloor` and `\rfloor` commands), the ceiling function ⌈ ⌉ (produced with `\lceil` and `\rceil`), the double vertical bars ‖ ‖ (used for the norm of a vector for example, and produced with the `\|` command).

An important property of delimiters in mathematical typesetting is that they must be of the right size to appropriately enclose the expression they delimit. So for example the output

$$\sin(\frac{1}{x})$$

is not acceptable in a properly prepared LaTeX document. The correct output needs to be

$$\sin\left(\frac{1}{x}\right).$$

LaTeX provides a convenient way to automatically adjust the size of any delimiter: place the command `\left` just before the left delimiter (with no spaces in between), and `\right` just before the right one. So the code
`$$f\left(\frac{1}{2}\right)$$` will produce the correct output for the last example.

Occasionally, we need to adjust the size of a right delimiter that does not have a matching left delimiter (more rarely, a left delimiter without a matching right delimiter). An example of the first case is an expression such as

$$\frac{t^2-1}{t^2+1}\Bigg|_0^1$$

that could occur in a Calculus problem. In this case, simply omitting the `\left` command will produce an error, because LaTeX expects each `\right` to have a corresponding `\left`. So we use an 'invisible' left delimiter with the command `\left.`, where the `\left` is followed by a period, with no spaces in between. So the code for the previous output is
`$$\left.\frac{t^2-1}{t^2+1}\right|_0^1$$`

Exercise 1. ⇒ Write code to produce the following equations:

$$\lim_{x\to 0}\left(\frac{\sin x}{x}-1\right)=0 \tag{1}$$

$$e=\sum_{n=0}^{\infty}\frac{1}{n!} \tag{2}$$

$$\frac{d}{dx}e^x=e^x \tag{3}$$

$$\frac{d}{dx}\int_0^x f(t)dt = f(x) \tag{4}$$

$$f(x) = \sum_{i=0}^{\infty} \frac{f^{(i)}(0)}{i!}x^i \tag{5}$$

$$x - \sqrt{x^2-1} = \frac{1}{x + \sqrt{x^2-1}} \tag{6}$$

$$\frac{t - \sin t}{1 + \sin t}\bigg|_0^{\pi/2} = \frac{\pi}{2} - 1 \tag{7}$$

If you need help, send email to your instructor, or look at the .tex file of this workbook.

5 Tables and matrices

5.1 The tabular command

The `tabular` command is used to typeset tables. By default, LaTeX tables are drawn without horizontal and vertical lines — you need to specify if you want lines drawn. LaTeX determines the width of the columns automatically.

This code starts a table:

```
\begin{tabular}{...}
```

Where the dots between the curly brackets are replaced by code defining the columns:

- `l` for a column of **left**-aligned text (letter el, *not* number one).
- `r` for a column of **right**-aligned text.
- `c` for a column of **centre**-aligned text.
- `|` for a vertical line.

For example, `{lll}` (i.e. left left left) will produce 3 columns of left-aligned text with no vertical lines , while `{|l|l|r|}` (i.e. |left|left|right|) will produce 3 columns — the first 2 are left-aligned, the third is right-aligned, and there are vertical lines around each column.

The table data follows the `\begin` command:

- `&` is placed between columns.
- `\\` is placed at the end of a row (to start a new one).
- `\hline` inserts a horizontal line.
- `\cline{1-2}` inserts a partial horizontal line between column 1 and column 2.

The command `\end{tabular}` finishes the table.

Examples of tabular code (on the left) and the resulting tables (on the right):

- A table with three rows and three columns and without lines. The three columns are aligned left, center, and right, respectively.

```
\begin{tabular}{lcr}
 10 & 20 & 30 \\
4 & 5 & 6\\
 700 & 800 & 900
\end{tabular}
```

10	20	30
4	5	6
700	800	900

- The same table with two vertical lines:

```
\begin{tabular}{l|c|r}
 10 & 20 & 30 \\
4 & 5 & 6\\
 700 & 800 & 900
\end{tabular}
```

10	20	30
4	5	6
700	800	900

- Now we add horizontal lines at the top and bottom:

```
\begin{tabular}{l|c|r}
\hline
 10 & 20 & 30 \\
4 & 5 & 6\\
 700 & 800 & 900
\hline
\end{tabular}
```

10	20	30
4	5	6
700	800	900

- This is the same table with all possible vertical and horizontal lines:

```
\begin{tabular}{|l|c|r|}
\hline
 10 & 20 & 30 \\
\hline
4 & 5 & 6\\
\hline
 700 & 800 & 900
\hline
\end{tabular}
```

10	20	30
4	5	6
700	800	900

- This is an example with a partial horizontal line:

```
\begin{tabular}{|r|l|}
\hline
 apple & green\\
banana & yellow \\ \cline{2-2}
 blackberry& black \\
 \hline \hline
tomato & red \\
\hline
\end{tabular}
```

apple	green
banana	yellow
blackberry	black
tomato	red

Exercise 2. Write code to produce the following tables:

1.

Item	Quantity	Price (in $)
Brush	3	5.50
Canvas	5	24.75
Oil paint	8	15.00

2.

	Year		
City	2016	2017	2018
New York	45789	49099	50023
Chicago	43087	45777	48891
Los Angeles	48677	47898	51234

5.2 The array command

The `tabular` command is used in text mode. If some entries of a table are math expressions, and we are using the `tabular` command, we need to use math mode for each of the entries with the math expressions. So for example in order to produce the table

Function	Domain	Range
$f(x)=x^2$	$(-\infty,\infty)$	$[0.\infty)$
$g(x)=\sqrt{x}$	$[0,\infty)$	$[0,\infty)$
$h(x)=x$	$(-\infty,\infty)$	$(-\infty,\infty)$

using the `tabular` command, we would need to use the code:

```
\begin{tabular}{c|c|c}
  Function & Domain & Range\\
  \hline
  $f(x)=x^2$ & $(-\infty,\infty)$ & $[0.\infty)$\\
  $g(x)=\sqrt{x}$ & $[0,\infty)$ & $[0,\infty)$\\
  $h(x)=x$ & $(-\infty,\infty)$ & $ (-\infty,\infty)$
  \end{tabular}
```

Note how many times we had to type `$`. But there is a better way: the `array` command is the equivalent of `tabular`. It is used in exactly the same way, but in math mode. This means that using `array`, the code for the previous table becomes:

```
$$\begin{array}{c|c|c}
  \mbox{Function} & \mbox{Domain} & \mbox{Range}\\
  \hline
  f(x)=x^2 & (-\infty,\infty) & [0.\infty)\\
  g(x)=\sqrt{x} & [0,\infty) & [0,\infty)\\
  h(x)=x & (-\infty,\infty) &  (-\infty,\infty)
  \end{array}$$
```

This time we did not have to use `$` for the many math entries. But we had to use `\mbox{ }` for the few text entries. So it should be clear what the best strategy is: if a table contains mostly text entries, use the `tabular` command, with math mode `$...$` for each of the few math entries. If instead the table contains mostly math entries, use the `array` command, with `\mbox{...}` for the few text entries.

⇒ Check that the code using the `array` command gives exactly the same output as the code using the `tabular` command

5.3 Matrices

A matrix is nothing but an array of (usually) numbers or other math expressions. So it can clearly be done using the `array` command. In fact, the matrix

$$\left(\begin{array}{cc} 0 & 1 \\ 1 & 1 \end{array} \right)$$

can be produced with the code

```
$$\left(\begin{array}{cc} 0 & 1 \\ 1 & 1 \end{array}\right)$$
```

But, there is a better way: if we add the package `\usepackage{amsmath}` in the preamble, the commands `\begin{pmatrix} ... \end{pmatrix}` (in math mode) will automatically insert the parentheses (so we do not need to type the `\left(` and `\right)` commands), and we do not need to know or keep track of how many columns the matrix is going to have. So the argument `{cc}` is not needed, and the above matrix can be produced with the simpler code

```
$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$
```

Exercise 3. Add `\usepackage{amsmath}` on a line before `\begin{document}`. Then write the code to produce the following output

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

5.4 Piecewise defined functions

A piecewise defined function such as

$$f(x) = \begin{cases} x^2 + 1 \text{ if } x \leq 2 \\ 1 - 3x \text{ if } x > 2 \end{cases}$$

is similar to a table or an array, and uses the commands `\begin{cases} ...\end{cases}`. The code to produce the above output is:

```
$$f(x) = \begin{cases} x^2+1 \mbox{ if } x\leq 2\\
1-3x \mbox{ if } x>2\end{cases}$$
```

Note the use of `\mbox` in order to write the word 'if'.

6 Figures

This section describes how to insert an image in your LaTeX document, which requires the `graphicx` package. Images should be PDF, PNG, JPEG or GIF files. The following code will insert an image called myimage:

```
\begin{figure}[h]
\centering
\includegraphics[width=1\textwidth]{myimage}
\caption{Here is my image}
\label{image-myimage}
\end{figure}
```

`[h]` is the placement specifier. `h` means put the figure approximately here (if it will fit). Other options are `t` (at the top of the page), `b` (at the bottom of the page) and `p` (on a separate page for figures). You can also add `!`, which overrides the rule LaTeX uses for choosing where to put the figure, and makes it more likely it will put it where you want (even if it doesn't look so good).

`\centering` centers the image on the page, if not used images are left-aligned by default. It's a good idea to use this as the figure captions are centerd.

`\includegraphics{...}` is the command that actually puts the image in your document. The image file should be saved in the same folder as the .tex file.

`[width=1\textwidth]` is an optional command that specifies the width of the picture - in this case the same width as the text. The width could also be given in centimeters (cm). You could also use `[scale=0.5]` which scales the image by the desired factor, in this case reducing by half.

`\caption{...}` defines a caption for the figure. If this is used LaTeX will add "Figure" and a number before the caption. If you use captions, you can use `\listoffigures` to create a table of figures in a similar way to the table of contents (section 2.6, page 6).

`\label{...}` creates a label to allow you to refer to the table or figure in your text (section 2.5, page 6).

Exercise 4. ⇒ Add `\usepackage{graphicx}` in the preamble of your document (before the `\begin{document}` command).

⇒ Upload an image file to your CoCalc account, in the same directory as the .tex file.

⇒ Type the following text at the point where you want your image inserted:

```
\begin{figure}[h!]
\centering
\includegraphics[width=1\textwidth]{ImageFilename}
\caption{My test image}
\end{figure}
```

Replace `ImageFilename` with the name of your image file, excluding the file extension. If there are any spaces in the file name enclose it in quotation marks, for example `"screen 20"`.

⇒ Click on **Build** button and check the PDF.

7 Inserting References

LaTeX includes features that allow you to easily cite references and create bibliographies in your document. The code

```
\begin{thebibliography}{1}
  \bibitem{Sage}  J. Harris, K. Kohl, J. Perry,
  \textit{Peering into Mathematics through Sage-colored Glasses},
  Lulu.com, 2016.
  \bibitem{LM} L. Lamport, \textit{\LaTeX\ a document preparation
  system: user's guide and reference manual},
  Addison-Wesley Longman Publishing Co., Inc. Boston, MA,
  USA \copyright 1994
\end{thebibliography}
```

will produce the following output at the end of your document:

References

[1] J. Harris, K. Kohl, J. Perry, *Peering into Mathematics through Sage-colored Glasses*, Lulu.com, 2016.

[2] L. Lamport, *LaTeX a document preparation system: user's guide and reference manual*, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA ©1994

In `\bibitem{Sage}` the curly braces contain the label used to reference the book, and the command to insert the citation is `\cite{Sage}`. So the code

```
The book \cite{Sage} provides a good introduction
to the computer algebra system SAGE.
```

will produce the output

The book [1] provides a good introduction to the computer algebra system SAGE.

and the code

```
Leslie Sampert's manual \cite{LM} is time tested
and authoritative reference for \LaTeX.
```

will produce the output: Leslie Sampert's manual [2] is a time tested and authoritative reference for LaTeX.

To include a page number in your in-text citation put it in square brackets before the citation key: `\cite[p. 215]{Sage}` produces [1, p. 215].

To cite multiple references include all the citation keys within the curly brackets separated by commas: `\cite{Sage, LM}` produces [1, 2].

7.1 Exercise

⇒ Create a bibliography in your document that lists two of your textbooks from this (or last) semester.

⇒ Add some content in your document that references the two books, for example you can write

```
The book \cite{book1} was hard to understand,
and the book \cite{book2} was too heavy to carry
```

where `book1` and `book2` are the labels you chose for the two books.

⇒ Click on **Build** and check the PDF file.

8 Further Reading

LaTeX Project
`http://www.latex-project.org/`
Official website - has links to documentation, information about installing LaTeX on your own computer, and information about where to look for help.

The Not So Short Introduction to LaTeX2e
`http://ctan.tug.org/tex-archive/info/lshort/english/lshort.pdf`
A good tutorial for beginners.

LaTeX Wikibook
`http://en.wikibooks.org/wiki/LaTeX/`

Comprehensive and clearly written, although still a work in progress. A downloadable PDF is also available.

Comparison of TEX Editors on Wikipedia
`http://en.wikipedia.org/wiki/Comparison_of_TeX_editors`
Information to help you to choose which LATEX editor to install on your own computer.

TeX Live
`http://www.tug.org/texlive/`
"An easy way to get up and running with the TeX document production system". Available for Unix and Windows (links to MacTeX for MacOSX users). Includes the TeXworks editor.

Workbook Source Files
The .tex file that was used to produce this workbook is in your CoCalc account.