Appendix A

LaTeX Information

A.1 Introduction

LaTeX is a very useful tool for making your lab reports look great, for this class and future ones. This appendix contains information about creating LaTeX documents, useful commands, and processing files. Refer to Appendix C for information on creating tables and inserting figures. When you see text written in `typewriter` font, that means you can type it directly as you see it written here.

A.2 Setting up the Document

A.2.1 Using $\mathcal{AMS}$-TeX

The $\mathcal{AMS}$-TeX package is included in OIT's basic LaTeX suite. This package, from the American Mathematical Society, contains several commands that make typesetting mathematics easier. While this guide will not always indicate which tools are from this package, you should note that some distributions of LaTeX must have the $\mathcal{AMS}$-TeX package installed separately.

A.2.2 Creating a LaTeX Document

LaTeX runs by reading code that has been typed into a text editor. You will be creating a text document that contains all the code LaTeX needs to process the document. You can use any text editor you want (pico, vi, emacs, etc.) to create run LaTeX, though in EGR53 we will primarily be using emacs.

A LaTeX file **must** be saved with the ending of `.tex`. As an example, let's say we want to name our file `lab1`. In other words, the *base* name of the file (i.e. the file name without any of the extensions) will be `lab1`. To open emacs and create this file, in the `xterm` window type next to your acpub ID type:

```
emacs lab1.tex &
```

and press enter. The purpose of the "&" is to allow you to continue typing into the xterm window. This will become helpful later when you are ready to view your document.

If you want to open your file again later, use the same command as above: emacs followed by the name of the file ending in .tex. Make sure you are in the right UNIX directory.

A.2.3 Command Basics

When Emacs opens, you will see a blank page. Now you can begin typing your commands. Most of the code in LaTeX consists of a series of commands which consist of a backslash (\) followed by a word, and often another word in brackets ({ }).When you type a command, or even just a backslash, it may appear in a different color of font style. Emacs has some helpful formatting commands that work with LaTeX code.

When the command includes a backslash followed by "begin", we say that we have initiated an *environment*. An environment is treated differently than the other text, depending on what you tell it to do. An environment is terminated with backslash followed by "end." For example:

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

Other commands simply include a backslash and a symbol or word:

```
\LaTeX \Delta \&
```

Note that in order to leave a space after the LaTeX symbol, you need to use the "space" command, which is simply a slash followed by a space. Punctuation that comes after LaTeX, however, works fine for LaTeX. The previous sentences, for example, were generated by:

```
Note that in order to leave a space after the \LaTeX\ symbol, you need
to use the ``space'' command, which is simply a slash followed by a space.
Punctuation that comes after \LaTeX, however, works fine for \LaTeX.
```

A.2.4 The Beginning and Ending of a LaTeX Document

To start a document, you need to give LaTeX some information. The first thing you must type on the page is the header. This allows LaTeX to recognize and run the document. The simplest header indicates nothing more than what class of document you are creating and then tells LaTeX that the document is about to begin:

```
\documentclass{article}
\begin{document}
```

In order to certain commands or to change the overall look and feel of a document, however, there may be a much more involved header. For example:

```
\documentclass{article}
\usepackage{amsmath}    % load AMS-Math package
\usepackage{epsfig}     % allows PostScript files
\usepackage{listings}   % allows lstlisting environment
\usepackage{moreverb}   % allows listinginput environment
\usepackage{vmargin}    % allows better margins
\setpapersize{USletter} % sets the paper size
\setmarginsrb{1in}{0.5in}{1in}{0.2in}{12pt}{11mm}{0pt}{11mm}
%sets margins
\begin{document}
```

When you are finished adding the code required for a document, the last line should be:

```
\end{document}
```

This tells LaTeX that the document is done and it can stop processing the `.tex` file.

A.3 Frequently Used Commands

A.3.1 Cut and Paste from MATLAB: the Verbatim command

Many of your lab reports will require that you cut and paste .m files or diaries from MATLAB. To insert these files into your document, in Emacs under the File menu, click Insert File. Then type in the correct directory and file name that you want to insert. For example, to insert the code from the .m file "MyFile.m" which is in your EGR53 `lab2` directory, go to Insert File and type:

```
~/EGR53/lab2/MyFile.m
```

The text of the .m file will now appear in your document. However, the most important thing to remember is to surround this text with the "verbatim" command. In the line above the file you have pasted, type

```
\begin{verbatim}
```

In the line following your pasted text, type `\end{verbatim}`. This command will ensure that LaTeX will not start trying to read your MATLAB code!

A.3.2 Math Mode

To type most of the mathematical equations and formulas you see in your LaTeX book, such as Greek letters, arrows, relational symbols, and so on, you're going to need to be in **math mode**. Otherwise, LaTeX gets stuck and the symbol does not appear as you intended. In addition, **arrays** must be created in math mode. There are a couple of ways to get into math mode. If you have just a short amount of text, like a single letter, exponent or formula, it is easiest to surround the text with dollar signs ($). For example, consider the following sentence:

We calculated the stress as $\sigma = 1.6 \times 10^6$ MPa and the strain as $\epsilon = 1.2$.

This is how the code appeared:

```
We calculated the stress as $\sigma = 1.6 \times 10^6$ MPa
and the strain as $\epsilon = 1.2$.\\
```

You do not want to surround the whole sentence with the $ symbols, because it will appear like this:

$$We calculated the stress as \sigma = 1.6 \times 10^6 MPa and the strain as \epsilon = 1.2.$$

You can use the $ symbol to get into math mode for longer equations as well, but another option is to surround the text with \[and \]. These work the same way as the dollar signs- simply surround the text that you want in math mode with these symbols.

A.3.3 Math Environments

The following commands automatically start a math environment, so it is unnecessary to add \[or $.

- `\begin{eqnarray*}`
  `\end{eqnarray*}`

  To get the equations:

$$ax^2 + bx + c = 0$$
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

  you would need to have:

```
\begin{eqnarray*}
ax^2+bx+c=0\\
x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}
\end{eqnarray*}
```

  as a part of your `.tex` document. The `begin` command is used to start the `eqnarray` environment which accepts an array of equations. The asterisk is an indicator to LaTeX **not** to number each line.

- `\begin{align}`
  `\end{align}`

  Using the $\mathcal{AMS}$-TeX package, you can line equations up with the `align` environment. You will need to put an & before the character that you aligned. Example:

$$ax^2 + bx + c = 0 \tag{A.1}$$
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{A.2}$$

  This is generated using:

```
\begin{align}
ax^2+bx+c&=0\\                    % <-- note location of &
x&=\frac{-b\pm\sqrt{b^2-4ac}}{2a}
\end{align}
```

  The equations are lined up on whatever comes after the ampersand and the equations are numbered (no asterisk after `align`).

- `\begin{align*}`
  `\end{align*}`

  This time the * suppresses numbering of the equations. Example:

$$3x + 4y = 10$$
$$a(x - x_0) + b(y - y_0) + c(z - z_0) = 0$$

  The code for this was:

```
\begin{align*}
3x + 4y &= 10\\
a(x-x_{0}) +b(y-y_{0})+c(z-z_{0})&=0
\end{align*}
```

Both `align` environments will center the group of equations.

- Multiple equations per line
  You can include multiple equations on a single line in the `align` and `align*` environments, using & to separate the different equations. Example:

$$\pi = 3.14159... \qquad e = 2.71828... \tag{A.3}$$

$$i = \sqrt{-1} \qquad e^{\pi i} = -1 \tag{A.4}$$

The code for this was:

```
\begin{align}
\pi&=3.14159... & e&=2.71828...\\
i&=\sqrt{-1} & e^{\pi i}&=-1
\end{align}
```

A.3.4 General Math Commands

These require math mode:

- `\frac{a}{b}`

  Creates the fraction $\frac{a}{b}$; replace a and b with anything.

- `\mbox{text}`

  When you want text to appear normal within an equation, instead of using italics and math spacing.

- `\lim, \hat{a}, \vec{a}, \imath, \jmath, \sqrt{a}`

  A couple of other useful commands; you might be able to guess what they do. If not, check the index of your text.

- `_{subscript}, ^{superscript}`

  Subscripts and superscripts - the brackets are optional for single character subscripts and superscripts but required for multiple characters.

- `\left`A B `\right`C- used to put brackets of symbols A and C to the left and right of whatever B is. A and C do not have to be the same thing, but there does have to be a `right` for every `left`. The possible symbols are in Kopka and Daly. An additional symbol is the period (.), which will produce a blank - this is useful if you want to only have a delimiter on one side. One thing to be careful about is using curly brackets {} as delimiters. You must put a slash in front of them first. For example:

$$f(x) = \begin{cases} 0 & x < 0 \\ \text{undefined} & x = 0 \\ 1 & x > 0 \end{cases} \qquad g(x) = \left| \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \right| = -2$$

can be produced with the code:

```
\begin{align*}
f(x)&=\left\{                    % <-- note slash in front of curly bracket
\begin{array}{ll}
0 & x<0 \\
\mbox{undefined} & x=0\\
1 & x>0
\end{array}
\right.&                        % <-- note the dot which produces a blank
g(x)&=\left|
\left[
\begin{array}{cc}
1 & 2 \\
3 & 4
\end{array}
\right]
\right|&=-2
\end{align*}
```

The use of superscripts, subscripts, special symbols, and fractions are all covered in the Kopka and Daly book. The exercises given at the end of some of the articles are good for learning how to produce mathematical equations. Some of them, however, depend on knowledge of functions covered in earlier sections.

A.3.5 Additional Commands

This is a brief list of the commands you are going to need frequently when writing lab reports. In addition, the index and appendices of Kopka and Daly as well as many web sites are excellent resources.

- Spacing Commands
  - \\: The equivalent to the "return" key; two backslashes will move to the next line. You can also add a number after the backslash: `\\[0.5 cm]`. This will create a space of 0.5 cm between the two lines. Another way to create extra space between lines is ∼\\ .
  - If you want to insert a space between two words, use \ followed by a blank space, or simply type ∼ between the two words.
- Page Setup Commands
  - `\pagebreak` or `\newpage`: Moves to the next page
  - `\begin{center}, \end{center}`
    Centers text. You can probably predict what happens if you replace "center" with "left" or "right".
  - `\section*{Your Section Heading}`
    Allows you to create headings for your report, such as Introduction, Discussion, etc. The purpose of the * is to suppress the numbering of the headings.
  - `\begin{enumerate}, \begin{itemize}`
    These initiate a listing environment. "Enumerate" includes numbering, while "itemize" includes bullets and dashes. Each numbered or bulleted item will begin with the command `\item`. The lists can also be nested. Make sure you finish with `\end{enumerate}` or `\end{itemize}`.
- Changing the appearance of font
  - `{\Large }, {\small }, {\large }...`
    Refer to your text on the various options for changing the size of font. Place the text you want to size within brackets. To change back to the normal size, use `\normalsize`.

- `{\it }, {\sl }, {\bf }, {\tt }...`
  Again, the various styles of font are included in your text. Use `\textnormal` to change back to normal text.
- Note: Make sure the brackets are around the whole command, not just the words you want to change. This will save you from having to repeatedly type `textnormal` or `normalsize`. For example:

  Bold and *Italics* are good ways to emphasize text.

  The code for this was:

  ```
  {\bf Bold} and {\it Italics} are good ways to emphasize text.
  ```

A.4 Processing a File

Processing a LaTeX file can be tedious, but it is important not to skip any steps until you are more comfortable with how LaTeX works. Note that much of the information in this section is also included in the UNIX Appendix (Appendix A). You can create a paper copy from a LaTeX document as follows:

- **Create and Save** When you have typed in a couple of lines in your file `name.tex`, where `name` is any file name of your choosing, and you want to see how they look, click Save in emacs.
- **Run LaTeX** Return to the xterm window and run LaTeX on the file by typing `latex name.tex` (if no file called `name` exists, you can also just run `latex name`). For example, to process the file titled `lab1.tex`, type:

  ```
  latex lab1.tex
  ```

- **LaTeX*will* get stuck sometimes** You will see the terminal window run a number of lines. If it stops without returning you to the command prompt, there is an error in the document. You may see something like this:

  ```
  ! File ended while scanning use of \textnormal .
  <inserted text>
                  \par
  <*> latex guide

  ?
  ```

  This means that LaTeX was unhappy with something you typed in. Often it will provide the line number (L.29 for example), and you can use emacs to go to that line and try to find the problem. The message above means that a bracket was omitted at the end of a passage. In the terminal window, type `x` next to the question mark. This will make LaTeX stop running.

  If LaTeX quit because you didn't specify a file (i.e. you just typed `latex` at the command line with no input file) or because it could not find the file you were trying to process (for example, you typed `labb1.tex` instead of `lab1.tex`) hit "CTRL-d" to cancel out of LaTeX. If LaTeX stops for any other reason, type "CTRL-c" to make LaTeX bring up the question mark. You can then use the `x` to get out.

- **Fix your mistakes and try again** Return to the emacs window and try to correct your document. Remember to click Save after each change is made. Repeat the steps above until there is no error. You will see:

  ```
  Transcript written on lab1.log.
  ```

  This means that you were successful.

- **The .dvi file** LaTeX creates a device independent - or `dvi` - file. The `kdvi` program can create a graphical version of this file so you can edit your document without wasting paper. Be aware that there are some types of graphics the `kdvi` cannot properly display - this will come out of the printer properly, however.

- **Viewing the file with kdvi** Run the file using `kdvi name.dvi &` (again, if no file named `name` exists, `kdvi name`). In the current example, `name` would be `lab1`:

  ```
  kdvi lab1.dvi &
  ```

- **Correct errors** Be thorough in checking your document for mistakes. If you find errors, you can automatically make correct them in the emacs file since it is still open. Rerun LaTeX. Note that if `kdvi` is still running, it will always bring up the most current copy of the dvi file; you will not need to run a *new* `kdvi` unless you closed the previous one.

- **Printing** Most of the time, you can just print directly from `kdvi`. When you click the printer button, there will be a printer name selection at the top of the screen - you will want to choose `bf lp0` - that is the name for the ePrint queue.

- **Creating PostScript file** The printer does not understand `dvi` files, only PostScript files, and sometimes `kdvi` cannot translate the information properly. In those cases, the `dvips` program will convert a `dvi` file to a PostScript file. The default setting for the program, however, is to send the new PostScript file directly to the printer. You will want to edit your file one last time, so it is better to create a PostScript file, look at it again, then send it to the printer yourself. You can do this using the command `dvips -o newname name.dvi` (if no file called `name` exists, `dvips -o newname name`). The `-o` tells the program to produce an output PostScript file called `newname`. If the `-o newname` is omitted, the PostScript will be sent directly to the printer. If just the `newname` is omitted, the output file will be called `name.ps`.

Command	What it does
`dvips foo.dvi`	prints the dvi file
`dvips foo.dvi -o`	converts `foo.dvi` to postscript and creates a file called foo.ps
`dvips foo.dvi -o bar.ps`	converts `foo.dvi` to postscript and creates a file called bar.ps

- **Printing** Finally, when you actually want to make a hard copy of your file, you have several options. You can print from `kdvi`, you can let `dvips` do the work as in the first example above, you can use the `lpr` command on a postscript file that has been created by `dvips`, or you can use the `kghostview` postscript viewer to look at the file and then print it through there.

  To get something to print to your room, you need to use `dvips` to create a PostScript file, use a file transfer program to copy the file from your OIT account to your personal computer and use GhostView on your personal computer to send the PostScript file to your printer. Since most people have not installed postscript viewers on their computer, you can use the PDF format instead. To do this, replace the `dvips` step above with the `dvipdf` command. This will create a PDF file that Adobe Acrobat can read. You can then transfer this file to your personal computer using a file transfer program and open and print it using the Acrobat Reader.

A.5 Summary of LaTeX Process

The following table summarizes the process for a file called `lab1.tex` to generate a PostScript file called `lab1print.ps`:

Command	**Purpose**
emacs lab1.tex &	start text processing program
(write and save file)	save file within text processing program
latex lab1.tex	process LaTeX file - may need to run this three (or four!) times
kdvi lab1.dvi &	view processed LaTeX file
(print from kdvi)	you may be able to just print from here. If not ↓
dvips lab1.dvi -o lab1print.ps	create PostScript file
kghostview lab1print.ps	view PostScript file and print

If you end up using labels and references as described in Appendix B, you may actually have to run the LaTeX step three times! The first time LaTeX goes through and figures out what all the labels mean, the second time it can figure out where they are, and the third time it can replace the references with their labels. In fact, running LaTeX a fourth time makes sure that any changes in page numbers caused by entering the page numbers themselves will be taken into account.