

# An Introduction to L<sup>A</sup>T<sub>E</sub>X

Irfan Saif and Susan Fisher  
Boston University  
Office of Information Technology  
Version 2.0

September 19, 1996

## Abstract

This tutorial is for people at Boston University who need to use  $\LaTeX$  to produce documents that combine text, graphics, and mathematics. It assumes no prior knowledge of  $\TeX$  or  $\LaTeX$ . All of the examples and exercises in this document were executed on a UNIX computer system.

Conventions used in this document: Items which are denoted as exercises are actions for the reader to try. Items printed in **bold** are commands, filenames and keywords that are common to the community of people who use  $\LaTeX$  UNIX and X.

This document will be updated each semester to reflect any changes in the system. The latest version is available at the front office of Information Technology, 111 Cummington Street, first floor.



©1995 BOSTON UNIVERSITY

Permission is granted to make verbatim copies of this document provided copyright and attribution are maintained.

Boston University  
Office of Information Technology  
111 Cummington Street  
Boston, MA 02215

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	What is L <sup>A</sup> T <sub>E</sub> X?	3
1.2	Why Use L <sup>A</sup> T <sub>E</sub> X?	3
<b>2</b>	<b>Getting Started</b>	<b>4</b>
2.1	Step 1 – Typing the Text and Formatting Information	4
2.1.1	Using Emacs to Input Text into a Document	4
2.1.2	The Basic L <sup>A</sup> T <sub>E</sub> X commands	5
2.1.3	Comments	6
2.2	Step 2 – Running T <sub>E</sub> X to produce a DVI File	6
2.3	Step 3 – Converting the DVI file to Printable Format	7
2.4	Step 4 – Previewing the PostScript Document	8
2.5	Step 5 – Printing the Document	8
2.6	L <sup>A</sup> T <sub>E</sub> X Errors	8
2.6.1	Recovering from an Error	8
2.7	Exercises for Chapters 1 and 2	10
<b>3</b>	<b>Formatting Your Document</b>	<b>11</b>
3.1	Special Characters in L <sup>A</sup> T <sub>E</sub> X	11
3.2	Type Styles	11
3.2.1	Changing Type Styles	11
3.3	Title Page and Abstracts	12
3.4	Typing Paragraphs	13
3.5	Organizing your Document into Sections	13
3.5.1	Sectioning Commands	13
3.5.2	Table of Contents	14
3.6	Creating Lists	14
3.7	Exercises for Chapter 3	15
<b>4</b>	<b>Mathematical Formulas</b>	<b>16</b>
4.1	When Mathematics Can Be Used	16
4.1.1	Mathematics in Sentences	16
4.1.2	Mathematical Formulas as Equations	17
4.2	Commands to Produce Mathematical Characters	17
4.2.1	Superscripts and Subscripts	17
4.2.2	Fractions	17
4.2.3	Roots	18
4.2.4	Greek Letters	18
4.2.5	Math Symbols	18
4.3	Exercises for Chapter 4	19
<b>5</b>	<b>Inserting a PostScript Graphic into a L<sup>A</sup>T<sub>E</sub>X Document</b>	<b>20</b>
5.1	Generating a Graphic Using Mathematica	20
5.2	Inserting PostScript into L <sup>A</sup> T <sub>E</sub> X using psfig	21
5.2.1	Scaling a PostScript Figure	22
5.3	Processing the L <sup>A</sup> T <sub>E</sub> X file	22

5.4	Previewing the Inserted Figure . . . . .	22
5.5	Where to go From Here . . . . .	22
5.6	Exercises for Chapter 5 . . . . .	23
<b>6</b>	<b>Model solution to the exercises</b>	<b>24</b>

# Chapter 1

## Introduction

### 1.1 What is L<sup>A</sup>T<sub>E</sub>X?

L<sup>A</sup>T<sub>E</sub>X is a program which does document typesetting similar to what a publisher requires. It is based on another program called T<sub>E</sub>X which was written by Donald Knuth, a professor at Stanford University. L<sup>A</sup>T<sub>E</sub>X is a set of T<sub>E</sub>X macros which are generally thought to be simpler to learn than T<sub>E</sub>X. Although T<sub>E</sub>X must be installed on a computer in order to use L<sup>A</sup>T<sub>E</sub>X, you do not need to be familiar with T<sub>E</sub>X in order to use L<sup>A</sup>T<sub>E</sub>X successfully.

### 1.2 Why Use L<sup>A</sup>T<sub>E</sub>X?

Many people use L<sup>A</sup>T<sub>E</sub>X because it is capable of displaying characters which are not found on a computer keyboard. For example the characters  $\lambda$ ,  $\sum$ ,  $\neq$  are all easily generated using L<sup>A</sup>T<sub>E</sub>X. Because of these features L<sup>A</sup>T<sub>E</sub>X is often used for writing documents which contain mathematics.

Another reason people commonly use L<sup>A</sup>T<sub>E</sub>X is because it does typesetting, not word processing. Word processors are capable of editing a document and formatting text on a page, but typesetters like L<sup>A</sup>T<sub>E</sub>X do much more. They look at each character on a page and arrange it so that the text is properly spaced for easy reading. Typesetting features like  **Kerning** and  **ligatures** are implemented in L<sup>A</sup>T<sub>E</sub>X. These types of readability concerns are seldom dealt with in word processors, and because they are built into L<sup>A</sup>T<sub>E</sub>X it is capable of producing very high quality documents.

L<sup>A</sup>T<sub>E</sub>X is freely available on many different computer platforms and this feature has made it very popular in colleges and universities around the world. Many faculty and students in areas that involve computation have used L<sup>A</sup>T<sub>E</sub>X for years. If you are interested in getting a copy of L<sup>A</sup>T<sub>E</sub>X for yourself, it is currently available for free on many machines. A list of systems and the L<sup>A</sup>T<sub>E</sub>X implementation available for each follows.

- Macintosh - OzT<sub>E</sub>X
- VMS - L<sup>A</sup>T<sub>E</sub>X
- UNIX - L<sup>A</sup>T<sub>E</sub>X
- DOS - SBT<sub>E</sub>X and EmT<sub>E</sub>X
- Atari - L<sup>A</sup>T<sub>E</sub>X
- Tandy 6000 - L<sup>A</sup>T<sub>E</sub>X
- Tops20 - L<sup>A</sup>T<sub>E</sub>X
- Amiga - CommonT<sub>E</sub>X

# Chapter 2

## Getting Started

There are five steps to creating a document with  $\text{\LaTeX}$ . They are:

1. Typing the text and formatting information for a document
2. Running  $\text{\LaTeX}$  to produce a device independent (DVI) file
3. Previewing the device-independent document (This is optional)
4. Converting the device-independent file to printable format
5. Previewing the PostScript file (This is also optional)
6. Printing the document

This chapter will introduce you to each of these steps and get you started writing your first  $\text{\LaTeX}$  document.

### 2.1 Step 1 – Typing the Text and Formatting Information

A text editor like Emacs must be used to input the text of your document into a file called an **input file**. The name of this input file should end in the letters “.tex” so that  $\text{\LaTeX}$  and Emacs will be able to recognize it as an input file. The instructions that tell  $\text{\LaTeX}$  how to format the document are also included in this input file. These formatting instructions are called  $\text{\LaTeX}$  **commands**. The majority of this document contains information on the various formatting commands for  $\text{\LaTeX}$ . Before going any further, let’s begin by typing the basic information into a “.tex” file so that we can use  $\text{\LaTeX}$ .

#### 2.1.1 Using Emacs to Input Text into a Document

This document assumes you are using Emacs to type your text into a file. If Emacs isn’t available on your system or you aren’t familiar with how it works, any text editor (like vi or ed) will do. <sup>1</sup>

If you are using an X terminal to do your work, you should get into the habit of using **emacs** to do your editing. The X version of Emacs adds the ability for your mouse to move the cursor around on the screen. Xemacs also opens a separate window to work in. Whenever you start an X application in an xterm window you should add an ampersand (&) at the end of the command. Type the following command:

```
acs% emacs tutorial.tex &
```

---

<sup>1</sup> If you are interested in learning Emacs attend a tutorial offered by Information Technology or pick up the tutorial handout titled “An Introduction to Emacs” available at Information Technology, 111 Cummington Street.

A new X window should appear on your screen which looks similar to this:

```
-----Emacs:tutorial.tex(TeX)-----All-----Loading tex-mode...done
```

At the bottom of the Emacs window notice that Emacs already knows that you are going to be typing a  $\text{\TeX}$  document and has loaded “tex-mode”. Tex-mode adds some features that will make your editing easier.

The following is the minimum set of  $\text{\LaTeX}$  commands that must be inserted into any document in order for  $\text{\LaTeX}$  to work. Type these into your tutorial.tex file.

```
\documentstyle{article}
\begin{document}
This is where your text should be placed.
\end{document}
```

Save the changes you’ve just made to your new  $\text{\LaTeX}$  file but don’t exit the text editor as we’ll be returning to it shortly.

## 2.1.2 The Basic $\text{\LaTeX}$ commands

$\text{\LaTeX}$  documents consist of two regions. The top region is known as the preamble, and main text is known as the body. Each  $\text{\LaTeX}$  document must have a minimum of three lines. This section explains what they are and why they are necessary.

### Documentstyle

$\text{\LaTeX}$  is generally used to create four different styles of documents. An **article** is usually a relatively short paper and should be thought of as a journal or magazine article. A **report** is a larger document, for example something that would contain separate chapters of text. A **book** in which the pages are commonly printed on two sides of the page, is the third type of document usually created with  $\text{\LaTeX}$ , and the last one is a **letter**. The first line that we typed into the “tutorial.tex” file is:

```
\documentstyle{article}
```

This states that our document will be an article. If we wanted to create a report, book, or letter instead we would simply replace the word **article** with one of these other styles.

This is your first  $\text{\LaTeX}$  command. You will need a similar line at the top of any  $\text{\LaTeX}$  document you create to tell  $\text{\LaTeX}$  the style of your document. Note that the backslash character is used by  $\text{\LaTeX}$  to denote the  $\text{\LaTeX}$  commands in the file. Any sequence of text beginning with a backslash will be interpreted by  $\text{\LaTeX}$  as formatting information rather than the text of your document. When a  $\text{\LaTeX}$  command is followed by text in curly brackets { }, the text is interpreted as an option or argument to the  $\text{\LaTeX}$  command.

### Beginning and Ending the Document

The second line of our “tutorial.tex” file looks like this:

```
\begin{document}
```

Everything following this line is considered the body of the document. The final line of our first  $\text{\LaTeX}$  document looks like this:

```
\end{document}
```

These two lines are necessary because everything between the beginning and end is included as the text of your document. Between these lines is where you type what you want to see printed on the page. Note that because the words **begin** and **end** start with a backslash they are two  $\text{\LaTeX}$  commands. The word **document** is surrounded by curly brackets and this argument tells  $\text{\LaTeX}$  that what we are beginning and ending is our document. A **begin** and **end** combination like this one is called an **environment** and will be discussed in more detail later.

### 2.1.3 Comments

It is always useful to embed comments in your documents so that you are able to understand what you did at any given time. It is also helpful when others read your files for modification or learning purposes.

To add comments the `%` symbol is used, as in the following sample file:

```
% This file was created by Joe Bloggs
% on November 11, 1994.
% filename : test.tex
```

```
\documentstyle{article}
\begin{document}
```

```
Hi, this is a test.
```

```
\end{document}
```

## 2.2 Step 2 – Running $\text{\TeX}$ to produce a DVI File

Once you have created a text file, the next step is to run the  $\text{\LaTeX}$  program so that it can format your file. Type the following command:

```
acs% latex tutorial
```

(NOTE: Because our file ends in “.tex” we may omit this ending from the file name in the command above and  $\text{\LaTeX}$  will understand which file we are referring to.)

What should appear on the screen is the following:

```
This is TeX, C Version 3.14t3
(tutorial.tex
LaTeX Version 2.09 <7 Dec 1989>
(/usr/local/lib/tex/inputs/article.sty
Document Style 'article' <16 Mar 88>.
(/usr/local/lib/tex/inputs/art10.sty))
No file tutorial.aux.
[1] (tutorial.aux) )
Output written on tutorial.dvi (1 page, 264 bytes).
Transcript written on tutorial.log.
acs%
```

Many people use  $\text{\LaTeX}$  without understanding what this output describes. This section of the tutorial

explains what this output means so that you will have a better understanding of the various files that are included in L<sup>A</sup>T<sub>E</sub>X and the process that L<sup>A</sup>T<sub>E</sub>X uses when formatting your file.

As you can see from the first line of output, the L<sup>A</sup>T<sub>E</sub>X program is closely linked with the T<sub>E</sub>X program. This line tells you which version of T<sub>E</sub>X is installed on the system.

The next line begins with an open parenthesis. Whenever T<sub>E</sub>X prints an open parenthesis followed by a file name in the output it is stating that it has opened this file and begun reading it. When it finishes reading the file it will print a closed parenthesis.

The first thing that T<sub>E</sub>X discovers when it reads the file “tutorial.tex” is that this file is a L<sup>A</sup>T<sub>E</sub>X file. T<sub>E</sub>X learns this because of the `documentstyle` command which we typed at the top of the file. T<sub>E</sub>X calls the L<sup>A</sup>T<sub>E</sub>X program and prints the version number for it on the screen.

The next piece of information L<sup>A</sup>T<sub>E</sub>X reads is that this document is to be formatted as an article. L<sup>A</sup>T<sub>E</sub>X opens another file called `/usr/local/lib/tex/inputs/article.sty` which describes what an article should look like. This file includes information about page numbering, positioning of margins, title pages, etc. After printing the version number of that file, L<sup>A</sup>T<sub>E</sub>X reads a file called `/usr/local/lib/tex/inputs/art10.sty`. This file describes what an article should look like when it is printed with a 10 point font. (This is the default font size for L<sup>A</sup>T<sub>E</sub>X.) After reading that file L<sup>A</sup>T<sub>E</sub>X also finishes reading the `article.sty` file (as seen by the two closed parenthesis at the end of the sixth line).

The next line states that there is no `tutorial.aux` file and L<sup>A</sup>T<sub>E</sub>X will create one. This file is called an **auxiliary file** and it can contain optional information in L<sup>A</sup>T<sub>E</sub>X, like table of contents information or indexing information.

On the eighth line L<sup>A</sup>T<sub>E</sub>X prints a [1] which means that it has formatted the first page of our document. If there were more than one page this number would be followed by a [2], [3], etc.

When L<sup>A</sup>T<sub>E</sub>X finishes reading the `tutorial.tex` document it closes the `tutorial.aux` file and creates a new file called `tutorial.dvi`. This file is called the **device-independent** file. It now contains all of your text in a formatted version of the document. This “dot-dvi file”, as it is commonly referred to, can be transferred to any other computer, looked at with a previewer, or converted into a printable form. It can not be edited or concatenated because it is no longer plain text.

Lastly the file `tutorial.log` contains a log of your L<sup>A</sup>T<sub>E</sub>X session. This file contains information on how many resources were used when you ran L<sup>A</sup>T<sub>E</sub>X and if there were any errors on the screen they would also be printed here. You may look at this file if you are interested in this information.

## 2.3 Step 3 – Converting the DVI file to Printable Format

The command used to convert a file from a “.dvi” file to something that can be printed depends on the type of machine and printer available. Most machines at Boston University have PostScript printers and therefore you should probably use the following command:

```
acs% dvips tutorial.dvi
```

```
This is dvips 5.47 Copyright 1986-91 Radical Eye
Software 'TeX output 1992.06.16:1438' -> tutorial.ps
<tex.pro>. [1]
acs2%
```

This command creates a PostScript file called `tutorial.ps` which can be displayed with a PostScript previewer or sent to a printer. If you do not have a PostScript printer ask your system administrator how you should print your L<sup>A</sup>T<sub>E</sub>X files.

## 2.4 Step 4 – Previewing the PostScript Document

Previewing a document means to look at it on the screen as it will appear when you print it. This can only be done after  $\text{\LaTeX}$  has been run on your “.tex” file to produce a “.dvi” file, and then `dvips` has been run on the “.dvi” file to produce a “.ps” file. Previewing is useful because it allows you to check that your document has been formatted as you expected. Previewing is a wonderful tool because it frees us from needing to print rough drafts of our document and thus saves paper. Previewing can be done on any X terminal or workstation running a version of the X Window System using the following command:

```
acs% ghostview tutorial.ps &
```

Ghostview will open a new window on your screen and will display our document in the window. Many people use `Xdvi` instead of `Ghostview`. Although `Xdvi` also allows you to preview documents, it does not have the ability to display graphics.

If you do not have an X display then you can see your document on a VT100 style terminal using the command:

```
acs% dvi2tty tutorial.dvi
```

This will scroll your file on your terminal. It can not display your text exactly as it will appear when it is printed but it gives you a reasonable likeness.

## 2.5 Step 5 – Printing the Document

The command to use to print your document depends on the type of machine and printer available.

For UNIX PostScript printers at BU the following commands should be used.

For double-sided printing:

```
acs% lpr tutorial.ps
```

For single-sided printing:

```
acs% lpr -Ppubps tutorial.ps
```

Both of these commands will send your file to your default printer.

## 2.6 $\text{\LaTeX}$ Errors

In the last sections we quickly reviewed all of the steps necessary to produce your first  $\text{\LaTeX}$  document. Before you complete your first document you will probably make a typing mistake or incorrectly use a particular  $\text{\LaTeX}$  command (we all do!). If this should happen you will get a  $\text{\LaTeX}$  error. This section will explain how to deal with an error in a  $\text{\LaTeX}$  file.

### 2.6.1 Recovering from an Error

Here is a file which contains a  $\text{\LaTeX}$  error. Can you find the error?

```
\documentstyle{article}
\begin{doc}
This is where your text should be placed.
\end{document}
```

What follows is what L<sup>A</sup>T<sub>E</sub>X will display if we try to process this file.

```
acs % latex error.tex
```

```
This is TeX, C Version 3.14t3
error.tex
LaTeX Version 2.09 <7 Dec 1989>
(/usr/local/lib/tex/inputs/article.sty
Document Style 'article' <16 Mar 88>.
(/usr/local/lib/tex/inputs/art10.sty))
LaTeX error. See LaTeX manual for explanation.
Type H <return> for immediate help.
! Environment doc undefined.
\@latexerr ...for immediate help.}\errmessage {#1}

1.2 \begin{doc}

?
```

L<sup>A</sup>T<sub>E</sub>X begins normally but it stops when it sees the error. The message that is displayed on the screen explains what the error is. The error message always begins with an explanation point (!). In this example the error says:

```
! Environment doc undefined.
```

Next, L<sup>A</sup>T<sub>E</sub>X tells you which line of the text caused this error:

```
1.2 \begin{doc}
```

The numbers 1.2 refer to the first file of your document and the second line. It then prints the line on the screen for you to refer to.

Lastly it displays a question mark. It is now waiting for your response. If you type the **RETURN** key it will try to continue ignoring the line that contains the error. In some cases this will work, but when the error is a fundamental one it cannot continue. In our example the error is fundamental because the line that L<sup>A</sup>T<sub>E</sub>X would skip is telling it to begin the document. Without this line everything else in the file is unrecognizable to L<sup>A</sup>T<sub>E</sub>X. At the ? prompt you should type an “x”:

```
? x
```

This will exit L<sup>A</sup>T<sub>E</sub>X and will allow you to edit the .tex file and fix the error. L<sup>A</sup>T<sub>E</sub>X will print the following on the screen:

```
No pages of output. Transcript written on error.log.
```

In addition to using x, i.e. the exit mode, you may also use the recover mode. This is invoked by typing an *r* instead of an *x*. The recover mode allows you to go through the remainder of the document and see if there are any other errors that need attention.

If you change the line:

```
\begin{doc}
```

to

```
\begin{document}
```

in the `error.tex` file it will work perfectly.

## 2.7 Exercises for Chapters 1 and 2

1. Create a  $\text{\LaTeX}$  **report** file called `functions.tex` using Emacs. Include the three statements which must be in every  $\text{\LaTeX}$  document.
2. In the body of this document, type the following text:  
A quadratic function is any function that can be written in the form:
3. Save the file and run  $\text{\LaTeX}$  on the file `functions.tex` using the UNIX command:  
**acs% latex functions**  
If this results in any errors, read Section 11 of *Essential  $\text{\LaTeX}$* . When  $\text{\LaTeX}$  has successfully processed your input file, it will supply you with a file called `functions.dvi`.
4. Convert this to printable PostScript form using the UNIX command: **acs% dvips functions**
5. Preview the PostScript file using the command: **acs% ghostview functions.ps &**  
**Do not print the file at this point!**

## Chapter 3

# Formatting Your Document

### 3.1 Special Characters in L<sup>A</sup>T<sub>E</sub>X

In the first chapter we learned that a backslash is used by L<sup>A</sup>T<sub>E</sub>X to denote that what follows it is a L<sup>A</sup>T<sub>E</sub>X command. The backslash is called a **special character**. There are a few other characters on your keyboard which cannot be used in the text of your document because they all have special meanings in L<sup>A</sup>T<sub>E</sub>X. You should learn these early so that you don't use them accidentally. They are: #, \$, %, &, ~, \, {, and }.

The following is a chart which explains what to type in your `.tex` file if you want one of these characters to appear in your final document.

To produce this:    Type this in your document:

#	\#
\$	\\$
%	\%
&	\&
-	\_
\	\$\$\backslash\$
{	\{
}	\}

### 3.2 Type Styles

L<sup>A</sup>T<sub>E</sub>X has different type styles which allow you to change the appearance of your text. These styles are:

<b>Bold Text</b>	= bf
<i>Emphasized</i>	= em or it
San Serif	= sf
<i>Slanted</i>	= sl
SMALL CAPS	= sc
<b>Typewriter</b>	= tt
Roman	= rm

Emphasized text is similar to italicized text.

#### 3.2.1 Changing Type Styles

There are two commonly used methods for changing the style of your text. The first method is used if you want a large section of text like a paragraph, a page, or a chapter, to appear in a different type. At

the beginning of the section you use a `\begin` command with the abbreviation for the desired type style following it in curly brackets. The section is ended with a similar `\end` command. For example:

```
This text is not bold.
\begin{bf}
This text is bold.
\end{bf}
This text is also not bold.
```

Everything before and after the `begin` and `end` commands will not be bold text. Everything between these commands will be bold text.

The second method used to change the style of text is used for single words or phrases. Place the word or phrase between curly brackets and name the text style within the brackets, as in the next example:

```
This text is not typewriter. {\tt This text is typewriter.}
This text is also not typewriter.
```

It is important that the command looks like this:

```
These words are {\sf san serif.} Are these words san serif?
```

and **not** like this:

```
This text is \sf{san serif.} Are these words san serif?
```

as the second example will change **all** the text following this command, even what appears after the curly brackets.

You may substitute any of the text styles in the examples above by replacing the “bf”, “tt”, or “sf” with the appropriate abbreviation for a different text style.

### 3.3 Title Page and Abstracts

When writing a paper one of the first things needed is a title page. The book, report, and article **document styles** all have different title page and abstract styles. In an **article** the title and abstract will appear on the same page as the body of your text. In a **report** if you have an abstract it will appear on a separate page either before or after a title page. A **book** does not have an abstract, but does have a title page.

Regardless of the style of document you are writing, a title page generally contains the title of the document, its author’s name and the date of the publication. Here is a simple example:

```
\begin{document}
\title{An Introduction to LaTeX}
\author{Irfan Saif and Susan Fisher}
\date{January 1, 2001}
\maketitle
```

The title page information comes immediately after the `\begin{document}` command. The first three lines describe the title, author, and the date. The last line tells  $\text{\LaTeX}$  to actually print a title page. Without the `\maketitle` command you will not see any titles on your document.

If you have more than one author for a paper and you want the name of each to appear on a separate line of the title page you should use a command similar to the following:

```
\author{Fred Flinstone \\ Barney Rubble}
```

The `\\` command tells L<sup>A</sup>T<sub>E</sub>X to begin a new line.

If you want the date on your title page to always reflect the last date that you ran L<sup>A</sup>T<sub>E</sub>X on this file then you can use the following command and L<sup>A</sup>T<sub>E</sub>X will fill in the proper date for you:

```
\date{\today}
```

If your document should have an abstract you should use the following command:

```
\abstract{Text of Abstract Goes Here}
```

If this command is placed before the `\maketitle` command in a report the abstract will be the first page of your report. If it appears after the `\maketitle` command it will appear after your title page. In an article the abstract and the title appears on the same page.

## 3.4 Typing Paragraphs

You can type new paragraphs in L<sup>A</sup>T<sub>E</sub>X by leaving a blank line between paragraphs in your `.tex` file. For example:

```
This text is in the first paragraph
of my new LaTeX document.
```

```
This text is in the second paragraph
because of the above blank line.
```

## 3.5 Organizing your Document into Sections

Most reports and books are broken up into different sections and chapters. (This document is a good example of that!) L<sup>A</sup>T<sub>E</sub>X has built in commands for dividing your document into sections and L<sup>A</sup>T<sub>E</sub>X will automatically format and number these sections and chapters for you. L<sup>A</sup>T<sub>E</sub>X can also generate a table of contents if you need one based on the titles of your sections and chapters.

### 3.5.1 Sectioning Commands

The four sectioning commands are:

```
\chapter{The Chapter Title}
\section{This is the First Section of the Chapter}
\subsection{This is a Subsection of the Section}
\subsubsection{This is a Sub-Subsection}
```

Within the brackets you should type the name of the section or chapter as you want it to appear in the table of contents and at the top of the page. Each chapter is started on a separate page and given a number automatically. The sections in that chapter are numbered starting from 1. A subsection is numbered with a decimal point (the first one being 1.1) and a subsubsection has two decimal points (the first one being 1.1.1).

One notable exception to this is that an **article** cannot have chapters and therefore the `\chapter{}` command is not recognized when using the article document style.

### 3.5.2 Table of Contents

L<sup>A</sup>T<sub>E</sub>X uses sectioning commands to create a table of contents for you. The command:

```
\tableofcontents
```

is usually placed after `\maketitle` at the beginning of the document. This command will automatically generate a table of contents.

An important note here is that L<sup>A</sup>T<sub>E</sub>X must be run **twice** to get the table of contents numbering correct. The first time that L<sup>A</sup>T<sub>E</sub>X processes your file it calculates the page numbering for the sections and/or chapters of your document. The second time that L<sup>A</sup>T<sub>E</sub>X processes your file it inserts these page numbers and section numbers into the table of contents. If you change the number of pages that your document has and you only run L<sup>A</sup>T<sub>E</sub>X one time your table of contents will be inaccurate.

After reading all of the information up to this point you are ready to do **Exercise 2** which is listed at the end of this document.

## 3.6 Creating Lists

In L<sup>A</sup>T<sub>E</sub>X creating a list of items is simple. There are three types of lists built in to L<sup>A</sup>T<sub>E</sub>X:

1. `itemize`: bullets mark each item in the list
2. `enumerate`: numbered list of items
3. `description`: you make the label for each item in the list

An itemized list is a list of items where a bullet is placed at the beginning of each item in the list. For example:

Things to do this week:

- pay bills
- laundry
- go to the zoo

This list is created using the following commands:

```
Things to do this week:
\begin{itemize}
\item pay bills
\item laundry
\item go to the zoo
\end{itemize}
```

An enumerated list is a numbered list of items. For example:

Groceries to buy:

1. vegetables
2. pasta

3. laundry detergent

This list is created using the following commands:

```
Groceries to buy:
\begin{enumerate}
\item vegetables
\item pasta
\item laundry detergent
\end{enumerate}
```

A description list is one where you make the label for each item in the list. For example:

**computer** Something that makes my life miserable.

**network** The aggregate of communications software and hardware that never work.

This list is created using the following commands:

```
\begin{description}
\item[computer]{Something that makes my life miserable.}
\item[network]{The aggregate of communications software and hardware
that never work.}
\end{description}
```

### 3.7 Exercises for Chapter 3

1. Edit `functions.tex` and make the words `quadratic function` appear in **bold**.
2. Add a title page to this file using the following information:  
Title: A Review of Functions    Author: use your name    Date: use today's date
3. Add a second paragraph to the body of this text which says:  
A polynomial function is a function that can be written in the form:
4. Add a third paragraph to the body of the text which says: A rational function is a quotient of two polynomial functions. Thus, the general form of a rational function is: (The format of these functions will be added in a later exercise).
5. Make each paragraph a separate chapter of your document using the “chapter” command. Name the first chapter **Quadratic Functions**, the second chapter **Polynomial Functions**, and the third chapter **Rational Functions**.
6. Add the command to have  $\text{\LaTeX}$  generate a table of contents on its own page.
7. Save your changes and rerun  $\text{\LaTeX}$  twice on this file. Preview it again using `ghostview`.

# Chapter 4

## Mathematical Formulas

Most people use  $\text{\LaTeX}$  because it is capable of displaying many mathematical characters in printed form which are not easily typed on a keyboard. This chapter will explain how to start using mathematics in your text.

### 4.1 When Mathematics Can Be Used

Before you start learning how to type mathematics using  $\text{\LaTeX}$  you need to know in which environments you can use these new commands. This section describes the different math environments. You can type mathematics only in these environments.

#### 4.1.1 Mathematics in Sentences

When formulas are included in a sentence, they are enclosed by the  $\text{\(}$  and  $\text{\)}$  commands, for example:

Does  $(x + y)$  always equal  $(y + x)$ ?

These symbols let  $\text{\LaTeX}$  know that what comes between them should be formatted as mathematics. This text, when processed by  $\text{\LaTeX}$  looks like this:

Does  $x + y$  always equal  $y + x$ ?

You will notice that the “x” and “y” are displayed in a different text style than the rest of the letters in the sentence because they are mathematical variables . This is done automatically when you are in a mathematics environment.

Here are two other ways to denote the same mathematical formula:

Does  $\$x + y\$$  always equal  $\$y + x\$$ ?

```
Does
\begin{math}
x + y
\end{math}
always equal
\begin{math}
y + x
\end{math}?
```

Both of these methods produce the same output as the first example. The dollar signs surrounding the

mathematics is meant to be used as a short-cut and the **begin** and **end** statements can be used when you wish to separate particularly lengthy math formulas in your  $\text{\LaTeX}$  file to make it more readable. The third method is unnecessary for our simple example.

### 4.1.2 Mathematical Formulas as Equations

In  $\text{\LaTeX}$  equations are formulas which appear on a separate line, not embedded by text. Each equation in a  $\text{\LaTeX}$  document is assigned a unique number. To write an equation you would use a set of commands like this:

```
\begin{equation}
2 x + y = 3
\end{equation}
```

To produce an unnumbered equation in  $\text{\LaTeX}$  enclose the equation between brackets  $\backslash[$  and  $\backslash]$  like this:

```
\[ 2x + y = 3 \]
```

which will produce the following:

$$2x + y = 3$$

## 4.2 Commands to Produce Mathematical Characters

The following sections will introduce you to some basic  $\text{\LaTeX}$  commands which produce math characters which aren't displayed on your keyboard. These commands can *only* be used within one of the math environments that were introduced in the previous section and all of the examples in this section demonstrate the various environments.

### 4.2.1 Superscripts and Subscripts

To produce a superscripted or subscripted numeral you must tell  $\text{\LaTeX}$  which numbers should be superscripted or subscripted.  $\text{\LaTeX}$  will automatically move these numbers to the appropriate locations and will print them in a smaller font size. In an equation a **superscript** is denoted by a caret (the shift-6 character),  $\wedge$ . In an equation a **subscript** is denoted by an underscore character,  $_$ . Superscripts and subscripts must be enclosed in curly brackets  $\{\}$  if they are more than one character long. The following are three examples that explain why:

$\backslash( x_2 + y^3 \backslash)$  will produce:  $x_2 + y^3$

$\backslash( 2^{\{23\}} \backslash)$  will produce:  $2^{23}$

$\backslash( 2^{\wedge}23 \backslash)$  will produce:  $2^23$

### 4.2.2 Fractions

Fractions are produced with the  $\backslashfrac$  command in the math environment.  $\backslashfrac$  takes two arguments, the numerator and the denominator. Both must be enclosed in curly brackets  $\{\}$ . Here are two examples:

$\backslashfrac{\{2\}}{\{3\}}$

will produce:  $\frac{2}{3}$

```
\begin{math}
\frac{x^2}{y_3}
\end{math}
```

will produce:  $\frac{x^2}{y_3}$

### 4.2.3 Roots

The command for producing roots is `\sqrt`. The default root is a square root. If you want to print a root other than a square root, place that number after the command in square brackets []. The number that you are taking the root of is placed in curly brackets {} after the command. The following are two examples.

The square root of 2 could be formed using the command: `$$\sqrt{2}$$`. This will produce:  $\sqrt{2}$

The cube root of 600 could be formed using the command: `\[ \sqrt[3]{600} \]`. This will produce an unnumbered equation:

$$\sqrt[3]{600}$$

### 4.2.4 Greek Letters

All the lowercase Greek letters and a subset of commonly used uppercase letters are available in the math environment. Here are some examples:

```
$$\pi$      =  $\pi$ 
$$\beta$    =  $\beta$ 
$$\Delta$  =  $\Delta$ 
```

### 4.2.5 Math Symbols

There are hundreds of math symbols defined in  $\text{\LaTeX}$ 's math environment. Rather than reproducing them all here I suggest you look on pages 43-46 of the  $\text{\LaTeX}$ User's Guide which is written by Leslie Lamport, the author of  $\text{\LaTeX}$ . Some examples of these characters are:

```
$$\int$    =  $\int$ 
$$\neq$    =  $\neq$ 
$$\sum$    =  $\sum$ 
```

### 4.3 Exercises for Chapter 4

1. In the **quadratic function** section of the file `functions.tex` add the following equation using the unnumbered equation environment:  $f(x) = Ax^2 + Bx + C$ .
2. In the **polynomial function** section add the following equation using the equation environment:  
 $f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$  (Notice how  $\text{\LaTeX}$  gives the equation a number.)
3. In the **rational function** section add the equation:  $f(x) = \frac{a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0}{b_m x^m + b_{m-1} x^{m-1} + \dots + b_1 x + b_0}$
4. In the **quadratic function** section of `functions.tex` make an itemized list which looks like this:  
Here is a list of two quadratic functions:
  - $f(x) = \frac{1}{4}x^2 - x$
  - $f(x) = 4x^2 + 2x + 3$
5. Save your changes and rerun  $\text{\LaTeX}$  on this file. Preview it again using `ghostview`.

## Chapter 5

# Inserting a PostScript Graphic into a $\text{\LaTeX}$ Document

This chapter is intended to provide you with a method to insert graphics into a  $\text{\LaTeX}$  document. The first section describes how to generate a PostScript image from Mathematica. If you already have a PostScript file to use, skip to section 5.2.

### 5.1 Generating a Graphic Using Mathematica

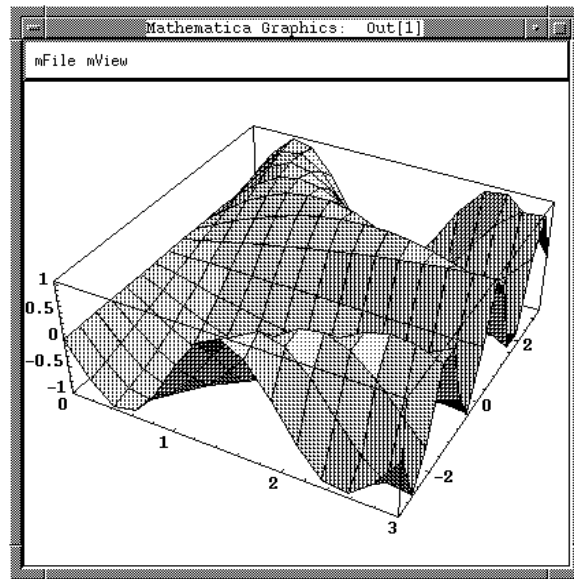
Mathematica is a package for doing various mathematical applications. One of the things it does is display three dimensional functions over a range values in the form of a surface graph. To generate a graphic in Mathematica you need to be using the X Window System. To start Mathematica type:

```
acs[username]% math
```

```
Mathematica 2.2 for IBM RISC System 6000  
Copyright 1988-92 Wolfram Research, Inc.  
-- Motif graphics initialized --
```

```
In[1]:= Plot3D[Sin[x y],{x,0,3},{y,-Pi,Pi}]
```

```
Out[1]= -SurfaceGraphics-
```



To save a graph to a file use the command:

```
In[2]:= Display["sin.mma",%]
```

```
Out[2]= -SurfaceGraphics-
```

To quit Mathematica, use:

```
In[3]:= Quit
```

To use the PostScript file **sin.mma** in L<sup>A</sup>T<sub>E</sub>X you need to first use the program **psfix** to modify the file so that the printer can understand it. The following command will convert the file **sin.mma** and save the resulting PostScript it **sin.ps**.

```
acs[username]% psfix sin.mma > sin.ps
```

The file **realsin.ps** can now be printed on any PostScript printer with the **lpr** command. It can also be included in a L<sup>A</sup>T<sub>E</sub>X file, as shown in the next section.

## 5.2 Inserting PostScript into L<sup>A</sup>T<sub>E</sub>X using psfig

**psfig** is a macro package which makes inserting PostScript files simple. At the top of the L<sup>A</sup>T<sub>E</sub>X file, after the line which says `\begin{document}` insert the **psfig** macro using this command:

```
\input{psfig}
```

Now you may use the **psfig** command anywhere in your L<sup>A</sup>T<sub>E</sub>X document to place your PostScript image on the page.

To insert a PostScript image into a page of a document, use the following L<sup>A</sup>T<sub>E</sub>X command:

```
\psfig{figure=<filename.ps>}
```

Where `<filename.ps>` is the name of the file which contains your PostScript figure. Using the example from the Mathematica section of this document, the command to insert this graph would be:

```
\psfig{figure=realsin.ps}
```

### 5.2.1 Scaling a PostScript Figure

There are options to `psfig` which allow you to enlarge or shrink the picture to fit into a certain space on the page. The options tell how wide and tall you want the picture to appear. To make the above example fit into a 2 inch x 2 inch square on the page, use:

```
\psfig{figure=realsin.ps,height=2in,width=2in}
```

Note that there are no spaces between the options to `psfig`. Use commas to separate the different arguments.

## 5.3 Processing the L<sup>A</sup>T<sub>E</sub>X file

Process your L<sup>A</sup>T<sub>E</sub>X file using the command:

```
acs[username]% latex <file>
```

When you run L<sup>A</sup>T<sub>E</sub>X on a file which has a PostScript figure in it you will see the following lines printed on the screen:

```
psfig: searching <file.ps> for bounding box
psfig: including <file.ps>
```

The bounding box tells L<sup>A</sup>T<sub>E</sub>X how much space to leave for the picture on the page.

## 5.4 Previewing the Inserted Figure

When you use a dvi previewer like `xdvi` you will not be able to see the PostScript figure in your document. This is because the PostScript file does not become a part of your paper until the entire L<sup>A</sup>T<sub>E</sub>X paper is converted to PostScript. When L<sup>A</sup>T<sub>E</sub>X converts your file into a .dvi file it only leaves a blank space for the figure to be inserted later. Most dvi previewers will print an error when they get to the stage where an image should appear on a page. Using a dvi previewer is still useful for previewing changes to the text of a document. They also display your paper more quickly because they do not spend time drawing the graphics on your screen.

Using a dvi to PostScript converter, like `dvips` will generate a printable form of your paper. This will also allow you to preview your document with a PostScript previewer like **GhostView**. When previewing at this stage your graphics will be included in the document.

```
acs[username]% dvips <file>
```

```
acs[username]% gs <file>.ps &
```

where `<file>.ps` is the printable form of your L<sup>A</sup>T<sub>E</sub>X file.

The file can be printed using the command:

```
acs[username]% lpr <file>.ps
```

## 5.5 Where to go From Here

1. For more information on **Mathematica**, Information Technology offers tutorials on the program. Handouts are also available at the front desk of Information Technology, first floor, 111 Cummington St.

2. For more information on **psfig**, see the document entitled *Psfif/TeX Users Guide*. This is written by the author of the macros and contains more information about inserting graphics. It is available via anonymous ftp from `whitechapel.media.mit.edu`.
3. For more information on **xdvi**, **ghostview**, or **gs** see the on-line manual pages. Type `man <command>` where `<command>` is the program you want to learn more about.

## 5.6 Exercises for Chapter 5

1. Start Mathematica using the UNIX command: `acs % math`
2. Use the following Mathematica command to generate a graph of the first quadratic function in the itemized list above: `Plot[Evaluate[1/4*x^2-x,{x,-3,7}]`
3. Use the following Mathematica command to save this graph to a file: `Display["quad.mma",%]`
4. Quit Mathematica and use the following command to convert the file to printable PostScript:  
`acs % psfix quad.mma > graph.ps`
5. Use the  $\text{\LaTeX}$  macro **psfig** to insert the graph into the **quadratic function** section of `functions.tex`. Make it 3 inches x 3 inches.
6. Save your changes and rerun  $\text{\LaTeX}$  on this file. Convert it to PostScript using **dvips**.
7. Preview it using **ghostview**.

## Chapter 6

# Model solution to the exercises

When all of the exercises are completed, the file should look like this:

```
\documentstyle{report}
\begin{document}
\input{psfig}
\title{A Review Of Functions}
\author{Your Name Here}
\date{\today}
\maketitle
\tableofcontents
\newpage
\chapter{Quadratic Functions}
A quadratic function is any function that can be written in the form:
\[f(x) = Ax^2+Bx+C\]
Here is a list of two quadratic functions:
\begin{itemize}
\item $ f(x) = \frac{1}{4}x^2 - x $
\item $f(x) = 4x^2 + 2x + 3$
\end{itemize}
\psfig{figure=graph.ps,height=3in,width=3in}
\chapter{Polynomial Functions}
A polynomial function is a function that can be written in the form:
\begin{equation}
f(x) = a_nx^n+a_{n-1}x^{n-1}+ \dots + a_1x + a_0
\end{equation}
\chapter{Rational Functions}
A rational function is a quotient of two polynomial functions. Thus, the
general form of a rational function is:
\begin{equation}
f(x)=\frac{a_nx^n + a_{n-1}x^{n-1} + \dots + a_1x + a_0}
{b_mx^m + b_{m-1}x^{m-1} + \dots + b_1x + b_0}
\end{equation}
\end{document}
```

# Index

- .aux file, 6
- .dvi file, 7
- .log file, 7
- TEX, 6
  
- abstract, 11
- Amiga, 3
- art10.sty, 6
- article, 5
- article.sty, 6
- Atari, 3
- author, 11
- auxiliary file, 6
  
- begin, 6
- bold text, 10
- book, 5
  
- chapter, 12
- chapters in articles, 12
  
- date, 11
- denominator, 16
- device-independent file, 7
- document, 6
- document styles, 11
- documentstyle, 5
- DOS, 3
- dvips, 7
  
- Emacs, 4
- emphasizing text, 10
- end, 6
- equations, 16
- errors, 8
  
- formatting instructions, 4
- fractions, 16
  
- input file, 4
- italicized text, 10
  
- letter, 5
- log file, 7
  
- Macintosh, 3
  
- numerator, 16
  
- PostScript, 8
  
- previewing, 7
- printing, 8
  
- reading files, 6
- report, 5
- roman text, 10
- roots, 17
  
- san serif text, 10
- section, 12
- section numbering, 12
- slanted text, 10
- small caps text, 10
- special characters, 10
- square root, 17
- subscript, 16
- subsection, 12
- subsubsection, 12
- superscript, 16
  
- Tandy 6000, 3
- tex-mode, 5
- title, 11
- title page, 11
- Tops20, 3
- type styles, 10
- typewriter text, 10
  
- undefined environment, 9
- UNIX, 3
- unnumbered equations, 16
  
- VMS, 3
- VT100, 7
  
- xdvi, 7
- xemacs, 4