Introduction to LaTeX

Lambert E. Murray

September 19, 2012

Abstract

This document describes the use of TEXMaker (a LATEX editor) and MikTEX (a version of LATEX written for windows) for developing input documents which utilize the TEX computer typesetting program to produce print-quality documents.

First you will be shown how to install and set up MikTEX and TEXMaker on your computer. You will then be introduced to the basic structure of a LATEX input document. Here you will be introduced to the way in which LATEX interprets the input file and produces the type-set output. You will learn some of the basic LATEX commands for inserting headings, footnotes, etc.

Next you will be shown how to input mathematical equations into your LATEX document, as well as chemical equations and symbols.

Finally, you will be shown how to form tables and how to input figures into your LATEX document.

Contents

Chapter 1

MikTeX and TeXMaker

LaTeX is a programming package that takes as its input simple ASCII text files, containing special LaTeX formatting commands. This program then uses the TeX typesetting engine to produce publication quality output, typically in the form of a .pdf file.

LaTeX has become the standard computer-based typesetting program used for electronic submission to journals within the scientific community. Each major publishing organization has developed their own unique document classes which are utilized by the LaTeX program. For example the American Physical Society (APS) has developed the APS REVTeX document class for all of the journals published by the APS. Each individual journal within the APS has its own unique options which are applied to this document class. For example, the command line:

```
\documentclass[twocolumn,
secnumarabic, amssymb, amsmath,
nofootinbib, tightenlines,
nobibnotes, showkeys, aps, prl]{revtex4-1}
```

specifies the revtex4.1 document class with several options in square brackets [ ], one of which is the "prl" option, shown above, which specifies formatting for Physical Review Letters. There are other options that include Physical Review A [pra], which covers atomic, molecular and optical physics; Physical Review B [prb], which covers condensed matter and materials physics; Physical Review C [prc], which covers nuclear physics; Physical Review D [prd], which covers particle physics, fields, gravitation and cosmology; Physical Review E [pre], which covers statistical, non-linear, and soft-matter physics; and Physical Review Special Topics [prstab] - both accelerators and beams (STAB) and educational research (STER).

Because of its wide-spread usage today in the research community, we feel that our students should be exposed to this powerful typesetting tool. Although LaTeX is somewhat overwhelming at first (primarily because the programming

environment is comprised of a large number of complex programs), several developments over the past few years have made it much simpler to use. There are now versions of LaTeX that run on the Windows operating system, and new text editors have been developed which helps to simplify the writing and editing of LaTeX documents.

One of the best distributions of LaTeX for Windows is MikTeX. You can download MikTeX for you own personal use from the MikTeX homepage,[1] and it's free! The MikTeX environment provides the capability of downloading only those LaTeX files that are necessary for start-up, and provides for *automatically* updating and upgrading your system as you need it!

The LaTeX editor that we will be using, TeXMaker, is a feature-rich integrated development environment (IDE) for developing LaTeX documents on Microsoft Windows. You will find that TeXMaker works seamlessly with MikTeX, and, best of all, TeXMaker is distributed free![2]

The aim of TeXMaker is to support the LaTeX newbie by providing him the most important LaTeX commands via menus and by making the use of the LaTeX compiler and other tools like MakeIndex and BibTeX and easy as possible. It also supports the LaTeX guru by providing a powerful, fully customizable integrated environment.

All of the tools needed to develop documents with LaTeX are integrated into TeXMaker. You have the editor to write your LaTeX files, menu bar buttons to build the output, error and warning flags in the compiler output to aid in source file debugging, and a navigation window that makes moving around the source file(s) a snap. Viewing the generated output is also easy with TeXMaker. The quick build menu item compiles and displays the output automatically in the built-in .pdf viewer included with TeXMaker.

1.1 Installing MikTeX and TeXMaker

If you go to my website[3] and click on the LaTeX link, you will find links to all the files you will need to set up and use LaTeX on your own personal computer. There are three sets of files that may be of interest:

- **Downloads**, containing the following programs that you will need to install on your computer:
  - MikTeX
  - TeXMaker
  - PrimoPDF
- **Reference Files**, containing a number of .pdf files that you may find useful in writing your LaTeX documents. The two most useful may be:

[1] http://miktex.org/2.9/setup
[2] http://www.xm1math.net/texmaker/download.html
[3] http://www.harding.edu/lmurray

  - "The Not So Short Introduction to LaTeX 2e" - One of the best I have found.
  - "Introduction to LaTeX" - My personal introduction to LaTeX.

- **Image Files**. These are example image files which may be useful for practice in including images into your LaTeX documents.

When you download and run the setup files from the "Downloads" section, be sure to first load MikTeX and, only then, TeXMaker, since TeXMaker will try to link to the main LaTeX program loaded on your computer.

1.1.1 Configuring MikTeX

Once you have installed MikTeXon you computer go to

```
Start>Programs>MikTeX 2.9>Maintenance>Package Manager
```

and click on the "Repository" button on the top menu. Choose "change package repository" and make sure that the radio button marked "Packages shall be installed from the Internet" has been selected. You will then be asked for the location of the repository that you wish to use. Choose the "USA ctan ...." repository. This tells MikTeX where to go to synchronize the files on your computer with the ones on the repository. To insure that the synchronization process works smoothly, go to

```
Start>Programs>MikTeX 2.9>Maintenance>Settings
```

and make sure MikTeX is set to automatically synchronize the files without having to ask your permission. Now, any files needing to be updated will be. In addition, if you specify additional packages for use in your LaTeX documents that are not on your computer, MikTeX will automatically go to the repository and retrieve them – placing all the necessary files in the appropriate folders.

1.1.2 Installing and Configuring TeXMaker

Once you have installed MikTeX, download and run the installation program for TeXMaker. TeXMaker should seamlessly link with the compiler without requiring any additional work on your part. Some LaTeX editors, when installed, will attempt to link to the default .pdf viewer on your computer (typically Adobe Acrobat Reader). This linking process should work flawlessly in the background. However, the two LaTeX editors that I have recommended (TeXWorks and TeXMaker) have their own .pdf viewer built in, which eliminates any potential problems that may arise with linking to an external viewer.

1.2 Your First LaTeX Document

You are now ready to use your LaTeX document development environment. In the TeXMaker window, click on the 'Wizard' menu button and select 'Quick

Start'. This opens a window which allows you to select certain options for your first document. Here you can select the 'document class' (choose article), the font size, enter your name as the author, and various other options. You can also select certain 'packages' to include in the header material of the document, such as the graphicx package (useful when you wish to include figures in your document), the geometry package which allows you to specify the margins, and the AMS package which allows you to typeset your mathematical formulae with the same high-quality typesetting that you might see in a mathematical journal (this one should always be selected). Once you have made your choices, you can click on 'ok' and the wizard with create the header material for your first LaTeX document. Do this now and examine the lines of text in the editor's open window. These are shown below:

```
\documentclass[10pt,letterpaper]{article}
\usepackage[utf8]{inputenc}
\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}
\usepackage{graphicx}
\usepackage[left=2cm,right=2cm,top=2cm,bottom=2cm]{geometry}
\author{Lambert E. Murray}
\begin{document}

\end{document}
```

The cursor will be located over a colored 'dot' between the `\begin{document}` and the `\end{document}` commands. This is where you will enter the text of your first document. The header material will be more thoroughly explained in the following chapter. But for now you should simply type the words "Hello World – I'm using LaTeX."

To see the output of the compiled document, go up to the menu bar and select 'Quick Build'. This will compile the document, and send it to the .pdf viewer, which should pop up automatically on your screen, if everything is working correctly. You can now examine both the .pdf output file and the "input" .tex file that produced it.

Chapter 2

Text Input File Structure

2.1 The LaTeX Command Structure

The first thing you must realize is that the LaTeX input document is a simple text document (which can be written in any common text editor) that contains special commands inserted to tell the LaTeX type-setting engine how to produce the proper output file. These type-setting commands are typically designated with the backslash symbol \, followed by a name consisting only of alphabetical characters. This command typically takes an argument which must be enclosed in curly braces { }, but may also take optional agruments which are enclosed in square brackets.

2.1.1 The Document Class

A good example of a typical LaTeX command is the command which specifies the type of file that you want to produce. Every LaTeX input file must begin with a LaTeX command which specifies the "document class" of the document. The document class is a pre-defined set of instructions that tells the type-setting engine how to set up the headings of the different sections (titles, abstracts, sections headings, etc.) - whether they are to be numbered, what font to use, etc. A typical document might begin with

```
\documentclass[10pt]{article}
```

as the first line of the text document. You can see that this follows the command structure of LaTeX: there is a backslash, an alphabetical command followed by an optional input (in square brackets), 10pt, which indicates the size of font, and the final argument specifies that the document class is *article* (not, for example, a book, or a report). Typically, an article is a much simpler document, with a title, author, and date on the same page as the text of the document itself, while a report would be a document with a separate title page, abstract page, table of contents, etc. Many journals have specific document classes that are

published for use with LaTeX. They typically have many optional settings that can be utilized to change different aspects of the formatting of the document.

2.1.2 The Overall Structure of the Document

The actual text of the document must be contained between the two commands

```
\begin{document}
\end{document}.
```

These actually set off the text "environment".

2.1.3 Other Environments

The Quote Environment Other enviroments can also be set up in your document. One that I have already used is the "quote" environment. This environment is contained between the two commands

```
\begin{quote}
\end{quote}.
```

This environment indents the text by an amount predetermined by the document class. In the examples above, you will notice that, in addition to being indented, the font of the indented material is different. This is due to the fact that I had to force the LaTeX document to print out exactly what I typed (with the backslash), rather then allowing it to "interpret" the LaTeX command which would actually type-set the text. This was done using yet another environment – the "verbatim" environment.

The Verbatim Environment To print out exactly what is typed, you make use of the the "verbatim" environment. This can be accomplished by enclosing the text between `\begin{verbatim}` and `\end{verbatim}`, or, within a paragraph by using the `\verb` command (a shortened version of the verbatim environment structure). Any text which is set off by some delimiting characters (which may be anything other than letters, * , or a space), and which follows the `\verb` command will be output just as you typed it! For example, you could type

```
\verb|\these   are   \not   \LaTeX commands|
```

which would produce `\these are \not \LaTeX commands`. You should notice that within the verbatim environment, you get "exactly" what you type. You control the spacing, the line breaks, etc. In this environment, the text is not automatically formatted to fit the page, it *will not wrap* as it automatically does for a LaTeX document!

Itemizing Environments It is often useful to include itemized lists within a document, which may be numbered or simply set off by some symbol. This is accomplished using an 'itemize', or 'enumerate' environment. These environments (and many others) can be easily inserted into your document using the menu tools within TEXMaker. Click on the 'LaTeX' menu at the top of the editor window and scroll down to 'List environment'. Here you can select

- the `\begin{itemize}` environment (which enters bullets), or
- the `\begin{enumerate}` environment (which enters numbered items)

You will notice that TEXMaker enters both the beginning and ending environment statements, and then places the cursor between these two, following the `\item` command – ready to accept your first item in the list. After entering the first item, you continue the list by entering another `\item` command, followed by the next item. In enumerated list, LATEX keeps up with the correct numbering so that you don't have to worry about it. If you later go back and eliminate an item, or add another item, the numbering is automatically changed.

2.2 Title, Author, Date, and Abstract

To create a title for your paper (along with author, date, and an abstract) you include the `\maketitle` command (without arguments) following the `\begin{document}` statement. The title, author, and date, however, must be set *before* this command is given. This is done by entering the following commands:

```
\title{This is the Title of the Document}
\author{Lambert E. Murray}
\date{\today}
```

where the argument of each of the commands is what will be output in the type-set document. The `\today` command enters the current day's date. If you want this to be a particular date, you just type that in.

Following the `\maketitle` command, you can create an abstract by using the "abstract" environment, just as we did the "quote" environment. In the "abstract" environment, the spacing, indentation, and font size are all predetermined by the document class that you specify.

2.3 Section and Paragraph Headings

As you will have noticed in this document, I have made use of section headings, subheadings, and paragraph headings. The section headings and subheadings are numbered, whereas the paraphraph headings are not. These are entered in the input document using LATEX commands of the form

```
\section{SectionName}
```

The numbering scheme, the fonts and styles, are all controled by the document class that you have specified for your document. The sectioning commands can be easily inserted into your document using the menu tools within TEXMaker. Click on the 'LaTeX' menu at the top of the editor window and scroll down to 'Sectioning'. Here you can select the type of heading (section, subsection, subsubsection, paragraph, etc.) and specify the name of the heading. TEXMaker will then insert the proper LATEX command. (You may want to include a label with a section name so that you can easily refer back to it later. This can be done using the 'label' command. More about this later.)

LATEX creates a table of contents by taking the section headings and page numbers generated from the previously compiled run of the document – thus a new document must be compiled twice to get a correct table of contents. All the sectioning commands mentioned above also exist as a "starred" version in the form

```
section*{...}
```

This form of the command generates section headings that are not numbered and that will *not* show up in the table of contents.

2.4 Useful Commands for Typsetting Text

2.4.1 White Spaces and Paragraphs

"Whitespace" characters, such as a blank or a tab, are treated uniformly as a "space" by LATEX. In fact, several *consecutive* whitespace characters are treated as *one* "space." Whitespace at the *start* of a line is generally ignored, and a *single* line break is treated as "whitespace." An empty line *between* two lines of text defines the end of a paragraph, in fact a *group* of empty lines is treated the same as *one* empty line. For example the input file

```
   You can enter one, or   several
spaces between words.
       You can even leave empty
space at the beginning of lines,
and LaTeX will format the document
properly.

To start a new paragraph, simply
leave an empty line (which serves to
signal an end of paragraph).


If you leave several empty lines the
result is the same.
```

when compiled by LATEX produces the following output

You can enter one, or several spaces between words. You can even leave empty space at the beginning of lines, and LaTeX will format the document properly.

To start a new paragraph, simply leave an empty line (which serves to signal an end of paragraph).

If you leave several empty lines the result is the same.

2.4.2 Printing Special Characters

There are a number of "reserved" characters in LaTeX. These characters typically have a special meaning or function under LaTeX. When the LaTeX text file is compiled these special characters are not printed as text, but often modify how the text is printed. We have already mentioned the use of the backslash as a special command character. These special characters are:

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

Notice that the square bracket is not a special character.

In order to have these special characters print properly as text characters within a document, we must tell LaTeX that these characters are *not* functioning as control characters in the text document. We have already looked at one way of doing this: the verbatim environment. Another approach is to use a special *control sequence* – the backslash (\) followed by the special character. As we have already pointed out, the backslash symbol begins all *control sequences*, or *commands* in a LaTeX document. Thus, typing the sequence:

```
\$ \& \% \# \_ \{ \} \~{} \^{}
```
[1]

in the LaTeX document produces the output

$ & % # _ { } ~ ^

when compiled. The backslash character, however, *cannot* be produced by typing "`\\`" because this is a reserved control sequence used to start a line break (*without* producing a new paragraph). In order to obtain the backslash as a text character in the output, you must enter the command "`\textbackslash`".

2.4.3 More on the LaTeX Command Sequence

As mentioned above, all LaTeX control sequences (or commands) begin with the backslash (\) symbol. These control sequences follow two formats:

- The backslash is followed by just one special character.
- The backslash is followed by the *name* of the control sequence. The name of the control sequence consists *only* of letters and is case sensitive.

[1] The double-braces following the symbols ~ and ^ are required for proper spacing.

The *named* control sequences may require additional {parameters} and/or [options] which immediately follow the name (with no intervening space). These control sequences are terminated by a space, a number, or any other "non-letter", and LaTeX ignores any whitespace *after* a named control sequence. In order to produce a space after a named control sequence you must enter either `{}` followed by a space, or you must enter a special spacing command. The control sequences for printing special characters, however, are treated like any other letter or symbol as far as spacing is concerned.

2.4.4 Comments: Documenting your LaTeX file

When LaTeX encounters a % character while processing the input file, it ignores the rest of the present line. This is especially useful in documenting a LaTeX input file, and I encourage you to document your files extensively – especially at the beginning, as you are learning to use LaTeX. These comments will help you as you discover different "quirks" about how LaTeX is sometimes "cajoled" into producing the output you desire.

2.4.5 Line and Page Breaks

LaTeX inserts the necessary line breaks, spaces between words, and hyphenations to optimize the words on a page and also make the output look good. However, occasionally things don't quite work out as you would like. This will often mean that you will need to insert a `\\` or `\newline` command, or a `\newpage` command. You can even specify how some words are hyphenated by using the `\hyphenation{`*word list*`}` command. For example, the command

```
\hyphenation{FORTRAN Hy-phen-a-tion}
```

shows how the word 'hyphenation' is to be hyphenated, and that the word FORTRAN is not to be hyphenated. Hyphenation problems, however, are rare unless you are using foreign terms that may not be in the LaTeX dictionary, or special terms that you do not want hyphenated. The 'hyphenation' command should be given in the preamble of the input file (i.e., before the `\begin{document}` command), and should contain only words built from normal letters. The word list in this command is *not* case sensitive.

2.4.6 Quotes, Dashes and Dots, and Emphasis

Special characters are often used in a document to draw attention to, or set off certain words or phrases. For example, we often use 'single' or "double" quotes to draw attention to a word, or we may *italicize* the word, or print the word in **boldface** type to draw attention to the word. These features are relatively simple to implement in LaTeX using standard LaTeX commands or special characters. To enclose a word in single quotes you simply enclose the word on the left with the ` symbol (or with the `\lq` command) and on the right

with the ’ symbol (or with the \rq command). For double quotes you simply repeat the character or command string. For example, if your input file is

```
You can type ‘single’ or ‘‘double’’
quotes by using the ‘quote’ keys, or
by using the \emph{command sequence}
for \lq single\rq{}or
\lq\lq double\rq\rq{}quotes.
```

the compiled output will be:

> You can type ‘single’ or “double” quotes by using the ‘quote’ keys, or by using the *command sequence* for ‘single’or “double”quotes.

In this last example, we made use of the \emph{text} command. The text editing environment of TEXMaker makes it easy to bring emphasis to a word. You can make the word **boldfaced**, or *italicized*, or *emphasized* by simply highlighting the word and selecting the button along the left side of the editor window with the **B**, the *i*, or the *e*. There are a number of other formatting menu buttons (as well as sectioning commands) which are available along the left side of the TEXMaker editor window. If you float the cursor over these you will get an indication of what each will do. You may want to try different ones and examine the output file to see what each one does.

2.4.7 Cross References and Footnotes

Using footnotes[2] in LATEX is especially easy, since LATEX keeps up with the footnote number automatically, and places the footnote at the appropriate place with the appropriate footnote font. All that is required is for you to enter the \footnote{text of footnote} command at the point where you want the footnote to appear. References are also fairly simple. You can refer back to any point in the document where you have placed a label. For example, we can refer back to the section entitled “Printing Special Characters” with the command \ref{sec:PrintingSpecialCharacters} to obtain:

> If you refer back to Section 2.4.2 you can see how to enter special characters.

You do have to type in the word “Section” before the reference command.

[2] This is a footnote.

Chapter 3

Typesetting Mathematics

When we first introduced the header material at the beginning of a typical LaTeX document (Section 1.2, page 5), you may have noticed the "usepackage" commands. These commands instructs the compiler to load additional features which are required for special tasks, for example the 'amsmath' and associated packages are used to properly typeset mathematical expressions. The "usepackage" command always follows the "document class" command. In more complex documents there may be a number of additional packages that must be added.

3.1 Equation Environments

There are two basic types of math environments: the in-line environment, and display environments. To produce in-line math, we use the dollar sign (`$`) to signal the LaTeX compiler that the enclosed text is to be formated as math. TeXMaker provides a menu button for entering in-line math (the $$ button on the left side of the editor window). For example, if you type:

```
Alber Einstein's famous equation $E=mc^2$ was developed
as a result of his special theory of relativity.
```

the LaTeX compiler will produce the following:

> Alber Einstein's famous equation $E = mc^2$ was developed as a result of his special theory of relativity.

The preferred manner of producing a "displayed" equation (i.e., one that is set off from other text by being centered on a display line) is to enclose a math expression within the command sequence `\[...\]`, as shown below:

$$E = mc^2$$

If you click on the math menu button at the top of the TeXMaker editor page, you will find a number of useful math environments that you can utilize (many

of which have short-cut key combinations). Some of these will be discussed below.

START HERE

3.2 Numbered Equations

You will often want the mathematical expressions to be numbered for easy reference. This can be accomplished using several different equation environments.

3.2.1 Single Displayed Equations

For displayed equations, it is preferable to use the `equation` and `equation*` environments for numbered and unnumbered, single-line equations, respectively as shown in the following examples. The first is a numbered equation:

$$a^2 + b^2 = c^2 \tag{3.1}$$

The next is an unnumbered equation:

$$y = mx + b$$

3.2.2 Multiple Aligned Equations

The `align` and `align*` environments are used for multiple (aligned) equations, all of which will be numbered (or unnumbered with the * version). You will need to tell the LaTeX compiler how to align the equations properly, and where you wish to break the equation. To tell the compiler how to align the equations, you must specify an alignment character in the equation body. If you use the preferred `\align` environment you need only use a "single" alignment symbol (`&`) *before* the equals sign, as shown in the following example. You will also have to tell the compiler where to separate the two equations (it's not smart enough to figure that out). For example you might type:

```
\begin{align}
a^2 + b^2 &= c^2\\
a^3 \times \frac {c^4}{a+b^3}
&= \sqrt[3]{d}+g^4
\end{align}
```

which would produce

$$a^2 + b^2 = c^2 \tag{3.2}$$

$$a^3 \times \frac{c^4}{a + b^3} = \sqrt[3]{d} + g^4 \tag{3.3}$$

3.2.3 Multi-Line Equations

Sometimes an equation is too long to fit on a single line. You use the `multiline` environment to typeset a single equation that will extend beyond a single line. Again, you will need to use the \\ symbol to indicate where to break the equation, as in the following example:

```
\begin{multline}
x_1+x_2+\ldots+x_n+1/x_1+1/x_2+
\ldots+1/x_n=\\
{y_1}^2+{y_2}^2+\ldots+{y_n}^2+
1/{y_1}^2+1/{y_2}^2+\ldots+
1/{y_n}^2
\end{multline}
```

which produces the following numbered equation

$$x_1 + x_2 + \ldots + x_n + 1/x_1 + 1/x_2 + \ldots + 1/x_n = \\ {y_1}^2 + {y_2}^2 + \ldots + {y_n}^2 + 1/{y_1}^2 + 1/{y_2}^2 + \ldots + 1/{y_n}^2 \quad (3.4)$$

Special Numbering Environments

Equations with Section Numbers If you want your equation numbers in the form Equation($m.n$), where m is the section number and n is the equation number you use the

```
\numberwithin{equation}{section}
```

command in the preamble of your document exactly as shown here (the arguments of this command are *not* variables.)

Subequations The `subequations` environment provides a convenient way to number equations in a group with a subordinate numbering scheme. For example

```
\begin{subequations}
. . .
\end{subequations}
```

causes all numbered equations within the "subequations" environment to be numbered $(4.9a), (4.9b), \ldots$, if the equation preceding the environment was numbered (4.8). By putting a `\label{grp}` command immediately following `\begin{subequations}` you can get a reference to the parent number. You can also label the individual subequations within this environment (see the example below).

The following example will serve to demonstrate the use of subequations:

```
\begin{subequations}\label{grp}
The equation
\begin{align}
Q_{ik} &= \sum_{j=1}^{N-1} x_{ij}
P_{jk}\label{grp1}\\
\intertext{and}
J_{ik} &= \prod_j x_{ij} P_{jk}
+ Q_{ik}\label{grp2}\\
\intertext{imply}
Q_{ik}-J_{ik} &= W_{ik}\label{grp3}
\end{align}
\end{subequations}
```

produces

The equation

$$Q_{ik} = \sum_{j=1}^{N-1} x_{ij} P_{jk} \tag{3.5a}$$

and

$$J_{ik} = \prod_{j} x_{ij} P_{jk} + Q_{ik} \tag{3.5b}$$

imply

$$Q_{ik} - J_{ik} = W_{ik} \tag{3.5c}$$

Sometimes when working with a list of equations a few words of text may make the sentence structure flow more smoothly. In the example above we used the `\intertext{}` command to enter text within the math environment.

Since we labeled this group of equations as Equation(3.5) we can refer to the second equation of this group as Equation(3.5b).

3.2.4 Boxed Equations

You may find a need to set a particular equation apart from others. It may have a particular signigicance, such as the final result of a long series of calculations. To do this you use the "boxed" environment as in the following simple example:

```
\begin{equation}
\boxed{E=mc^2}
\end{equation}
```

which would produce

$$\boxed{E = mc^2} \tag{3.6}$$

3.3 Tips on Entering Math

3.3.1 Spaces and Carriage Returns

When entering mathematical expressions, any extra spaces that you may place in an equation are typcially ignored when the text file is compiled – in fact, carriage returns are also completely ignored. For example, the different inputs

```
\begin{subequations}
\begin{align}
x ^ 2 + y ^ 2 + z ^ 2 &= 0\\
x^2+y^2+z^2&=0\\
x^2+
     y^2+
          z^2
              &=
                 0
\end{align}
\end{subequations}
```

all produce the same output

$$x^2 + y^2 + z^2 = 0 \tag{3.7a}$$
$$x^2 + y^2 + z^2 = 0 \tag{3.7b}$$
$$x^2 + y^2 + z^2 = 0 \tag{3.7c}$$

Notice that we did have to use the double backslash to tell the compiler where to end the equations, and that we used the alignment character to make sure the equations lined up properly. Even though I may be able to type a mathematical expression with no spaces, I may find it harder to read this equation in the input file. I fact, sometimes it is an advantage to be able to put spaces into the math expressions to help things "line" up and make them easier to read.

3.3.2 Subscripts and Superscripts

In math mode, the subscript indicator is the underscore character and the superscript indicator is the carat (the character above the 6). But how does the LaTeXcompiler know what to place in the superscript or subscript? In math mode all subscripts and superscripts are applied *only* to the next *single* character, unless you specifically group characters together using {}. The order of the subscript or superscript is also totally irrelevant. For example, consider the two different input text streams below:

```
\begin{align}
y&=\sum_{i=1}^{n}x^2_i=x_1^2+x_2^2+\dots+x_n^2
y&=\sim_{i=1}^nx_{i}^2=x_{1}^2+x_{2}^2+\dots+x_{n}^2
\end{align}
```

The LaTeX compiler produces the same output for both!

$$y = \sum_{i=1}^{n} x_i^2 = x_1^2 + x_2^2 + \cdots + x_n^2 \tag{3.8}$$

$$y = \sum_{i=1}^{n} x_i^2 = x_1^2 + x_2^2 + \cdots + x_n^2 \tag{3.9}$$

Two short-cut buttons are provided on the left side of the TeXMaker editor window to help you enter subscripts and superscripts. By the way, you can place the subscripts and superscripts both *before* and *after* text characters.

3.3.3 Chemical Reaction Equations and Symbols

Using TeXMaker's helpful buttons, or just the standard LaTeX input we can also construct nuclear reaction equations of the form:

$$_{92}U^{235} +_0 n^1 \rightarrow_{92} U^{236} \tag{3.10}$$

where one has to be careful to include within curly braces all the numbers that are to be super- or subscripted. The spacing in this last expression, however, does not look quite right. We need some space between the "plus" sign and the zero subscript before the n, and perhaps just after the right arrow. But we can't just add a space, because the compiler will ignore spaces in math mode. The answer to this issue is a LaTeX command sequence used to produce a "thin" space in math mode. This is the control sequence "`\,`." Inserting this thin space into our nuclear reaction expression now produces

$$_{92}U^{235} + {}_0n^1 \rightarrow {}_{92}U^{236} \tag{3.11}$$

which looks a little better.

We can also create expressions like

$$^{12}_{6}\mathrm{C} \qquad \text{versus} \qquad {}^{12}_{6}\mathrm{C} \tag{3.12}$$

with the code

```
\begin{equation}
^{12}_{\phantom{1}6}\textrm{C}
\qquad \textrm{versus} \qquad
^{12}_{6}\textrm{C}
\end{equation}
```

Notice the use of the `\phantom{}` command in this last example to control the spacing in the subscript. The phantom command basically enters the number of blank spaces that you specify. The size of the blank spaces is based upon the font size used in your document.

3.3.4 Entering Fractions

Fractions can be designated in a mathematical expression in one of several ways. You can enter the slash (/) between the numerator and denominator, or you can insert the `\div` command between the numerator and the denominator, or you can use the `\frac{numerator}{denominator}` command. For example

```
\[
(x+y)/z = (x+y)\div z = \frac{x+y}{z}
\]
```

produces

$$(x+y)/z = (x+y) \div z = \frac{x+y}{z}$$

Each of these different commands gives a different representation of the same mathematical expression for division. If we were to express each of these as an in-line equation, we would obtain $(x+y)/z = (x+y) \div z = \frac{x+y}{z}$. Notice that the stacked version (the last term) of this expression looks a bit squeezed! The displayed version of these equations were typeset in "display" mode, with the numerator and denominator of the stacked division each typeset in "text-sized" (or t-sized) print. When the equation is typed as an in-line expression, the stacked expression is in "text" mode (t-sized) which means the numerator and denominator are typeset in the smaller s-sized print (subscript size).

There are ways of overriding the automatic font-size adjustments, but these are not recommended since that also influence the line-spacing of the document.

3.3.5 Entering Integrals

To express integrals in LaTeX you use the `\int` command. This command takes upper and lower limits and is sized differently in displayed or in-line equations (as is `\sum` and other mathematical operators). For example the in-line version of the integral looks like $\int_1^5 f(x)\,dx$, while the displayed version looks like

$$\int_1^5 f(x)\,dx$$

LaTeX also has the special expression `\oint` for closed integrals, which produces

$$\oint g(x)\,dx$$

Sometimes you will need to express multiple integrals. You could type

```
\[ \int \int f(x,y) dx dy \]
```

which will produce

$$\int\int f(x,y)dxdy$$

This looks pretty good, but is not typeset quite as nicely as we would like: the integral symbols are really too far apart, and the spacing between the mathematical symbols need adjustment for clarity. The first problem is solved by using the special command `\iint` which indicates that we want two integral signs (`iiint` would designate 3, etc.). The second is accomplished by using the special 'thin' space command "`\,`" to add appropriate space between the mathematical symbols. Thus, if we type

```
\[ \iint f(x,y)\,dx\,dy \]
```

we obtain

$$\iint f(x,y)\,dx\,dy$$

which looks just right.

3.4 Spacing in Math Expressions

3.4.1 Horizontal Spacing

If the spaces chosen by LaTeX within a mathematical expression are not satisfactory, they can be adjusted by inserting special spacing commands. For small spaces, there are the commands: (`\,`) for 3/18 quad; (`\:`) for 4/18 quad; and (`\;`) for 5/18 quad, where the size of a 'quad' corresponds to the width of the character M of the current font. For larger spaces, we can use the (`\quad`) and (`\qquad`) commands. If we wish to 'close up' a formula, we can use the (`\!`) command which produces a *negative* space of −3/18 quad.

3.4.2 Vertical Spacing

Occasionally within displayed mathematics, or even in simple text, you might like to change the vertical spacing. For example, you might want to produce the following:

This is some text.

This is some more text.

And this is yet some more text.

This was produced with the following input:

```
This is some text\\[15pt]
This is some more text\\[30pt]
And this is yet some more text.
```

where `\\[15pt]` indicates a 15–point line break, etc. Font sizes in LaTeX are specified by typing a number followed immediately by a dimension, such as 'pt' or 'pc'. For example, the font size `[10pt]` stands for 10 points, where 1 in = 72.72 pt. Similarly, `[1pc]` stands for 1 pica unit, where 1 pc = 12 pt.

LaTeX has a large array of spacing capabilities, both horizontal and vertical, but the standard spacing within LaTeX will usually serve us quite well.

3.5 Creating New LaTeX Commands

When typing a document – especially mathematical documents – you may find yourself repeatedly entering the same block of text, or math symbols. To simplify your task you can create a new command which will do much of the work for you. For example, the control sequence `\LaTeX` prints LaTeX in the document. I used this so often that I decided to create the command `\lt` which would enter this control sequence for me (typing the upper and lower cases got old). However, correct spacing now becomes an issue.

You will recall that a named command such as `\LaTeX` is terminated by a space (or any other non-letter) – so a space is often required at the end of the command. However, an extra typed space is often not interpreted as a space for typesetting purposes, since the LaTeX compiler ignores all extra spaces between words and at the beginning of lines. This means that you sometimes have to insert an extra space after a command. To accomplish this, you insert a `{}` directly following the command in order to designate the "end" of command. Thus, although I defined the command

`\newcommand{\lt}{\LaTeX}` to typset LaTeX,

I have to insert `{}` following this command everywhere I need an additional space. I *could* have created the command

```
\newcommand{\lt}{\LaTeX{}}
```

which would automatically insert the space after typsetting LaTeX, but then I would not have had the opportunity to explain this little detail to you.

The form of the control sequence which defines a new command is:

`\newcommand{name}{command-sequence}`.

You can define as many of these as you wish and place them in the preamble of your document (before the `\begin{document}` command). (You can even put a user-defined command inside a user-defined command.) For typing this document, I have used the user-defined commands:

```
\newcommand{\lt}{\LaTeX}
\newcommand{\tm}{{{\TeX}Maker}}
\newcommand{\miktex}{Mik{\TeX}}
```

to simplify the typesetting of specialized text elements.

The real advantage of user-defined commands, however, is best seen in typesetting complex mathematical expressions. For example, the Schrödinger equation

$$-\frac{\hbar^2}{2m}\frac{\partial^2\Psi(x,t)}{\partial x^2} + V(x,t)\,\Psi(x,t) = i\hbar\frac{\partial\Psi(x,t)}{\partial t}$$

pops up quite often in Modern and Quantum Physics texts. The term on the left with the second partial derivative must be entered as

```
\frac{{\partial}^2\Psi(x,t)}{\partial{x}^2}
```

It would obviously be convenient to introduce some short-hand way of writing these partials without typing this each time. This can be quite easily accomplished using the `\newcommand` feature of LaTeX. We can define a second-partial operator which we will call `\ppx{}` with the following command:

```
\newcommand{\ppx}[1]
{\ensuremath{\frac{{\partial}^2 #1}{\partial{x}^2}}}
```

This command takes only one argument (as indicated by the [1]) but commands may take up to 9 arguments. The argument is inserted at the position designated by #1 within the definition statement of the command. So, if we type the command sequence `\ppx{}` with some expression within the braces, that expression is inserted into our definition. For example, if we type `\ppx{\Psi(x,t)}` we obtain

$$\frac{\partial^2\Psi(x,t)}{\partial x^2}$$

One can define other similar operator expressions, for producing all sorts of complicated expressions. We could even define the control sequence `\Sch` which would type out the entire (Sch)rödinger equation in one step! This is obviously what you would want to do if the equation comes up a lot in your document.

> A suggestion on procedure is in order here. You will soon discover that it is really easy to mess up a mathematical expression. When this occurs within a complex document, the errors you receive may become nearly impossible to decipher. It is often advisable to create a document which can be used to develop and test equations before you place them into a larger document. You should create a document such as 'MathTest.tex' and write your complex mathematical expressions in this document. When you have it in final form (with no errors) you can cut and paste to your main document. This document should contain the document class entry `\documentclass[amssymb, amsmath]{amsart}`. This document class has the appropriate basic settings for mathematical typesetting. (You may even need to include additional packages to augment this document class.)

Chapter 4

Creating Tables

You will often need to present your data in data tables. These can be a little challenging at first, but with the help of the tabular wizard in TEXMaker they are easily produced. Suppose you wanted to report the experimentally measured wavelengths of the visible spectral lines of Hydrogen, and that you wanted to record them according to the spectral order used in making the measurement. Table 1 is an example of what you might want to produce to present this data.

Table 4.1: Determination of Hydrogen Spectrum Wavelengths per Order

	Spectral Order		
Hydrogen Line	First	Second	Third
Red	657.0	656.7	656.3
Blue	487.3	486.7	486.1
Violet	433.5	434.3	()*

* Not visible in third order.

This table actually has three parts:

- a caption
- the actual table of information
- a footnote

To produce such a table we will make use of an extra feature which may require you to add a new package – the three-part-table package. But first, we will concentrate on producing the actual table of data without the caption and footnotes. This task is relatively easy if we use the TEXMaker wizard.

4.1 Creating Tables with TEXMaker's Table Wizard

To create tables using TEXMaker simply click on the 'Wizard' menu button and then click on 'Quick Tabular'. This will open the 'Quick Tabular' window. The top portion of this window gives you a template of the table you are creating. The default template has two rows and two columns. The bottom of the 'Quick Tabular' window allows you to modify the default template. You can

- change the number of rows and columns in your table
- choose the alignment of the elements within the cells of the table
- choose the type of borders for the cells in the table
- merge columns in different rows

Once you have formed the basic template for your table, you simply enter your data into the cells of the table template at the top of the window. Once you have the information entered into your table, just click the 'ok' button on the wizard and the editor will generate the LATEX code to create your table. But once the code is generated you cannot recall the wizard with your previous information – if you made a mistake, you will have to edit the LATEX code or start all over.

You should try to create the basic table of data shown above using the TEXMaker wizard. Don't worry about the caption (or title) for the table, or the footnote yet – we will show to do that later. Once you generate the table using the wizard and compile the LATEX document your .pdf output should look something like the following:

	Spectral Order		
Hydrogen Line	First	Second	Third
Red	657.0	656.7	656.3
Blue	487.3	486.7	486.1
Violet	433.5	434.3	()*

Look at the LATEX code that was generated by the TEXMaker wizard to produce this table. We will attempt to decipher this code in just a bit, but first, let's clean things up a bit and add the table caption and the footnote. You will typically want the table centered and offset somewhat from the rest of the text. This can be accomplished by enclosing the table in a centering environment, which will produce:

	Spectral Order		
Hydrogen Line	First	Second	Third
Red	657.0	656.7	656.3
Blue	487.3	486.7	486.1
Violet	433.5	434.3	()*

A mentioned above, a special LATEX environment has been developed to create tables with captions and foonotes. To utilize these features, you will

need to enclose the LaTeX code for your basic table within the following lines of LaTeX code:

```
\begin{table}[here]
\begin{center}
\begin{threeparttable}
\caption{Determination of
 Hydrogen Spectrum Wavelengths per Order}
\label{tab:HSpec}
TABULAR CODE GOES HERE
\begin{tablenotes}
\item[*]Not visible in third order.
\end{tablenotes}
\end{threeparttable}
\end{center}
\end{table}
```

This will create the following table with a caption and with the footnote.

Table 4.2: Determination of Hydrogen Spectrum Wavelengths per Order

	Spectral Order		
Hydrogen Line	First	Second	Third
Red	657.0	656.7	656.3
Blue	487.3	486.7	486.1
Violet	433.5	434.3	()*

* Not visible in third order.

The actual LaTeX code used to create the table at the beginning of this chapter is given below. It is very similar to the code generated by the TeXMaker wizard. However, slight differences occur which show up as slight differences in the appearance of the two tables (some vertical lines are not present in the table at the beginning of the chapter which are present in the table generated by TeXMaker). As you work your way through the code below, you should try and pick up on the differences in the two sets of code. (You will notice that the code generated by TeXMaker contains the `\multicolumn{}{}{}` command, whereas the code for the table at the beginning of this chapter contains the `\cline{-}` command. You can click on the 'Help' menu button in TeXMaker and select 'LaTeX Reference' to look up information on these two commands. You should look up 'tabular' first which will give you an overview of the tabular commands for forming a table – then you can look up the 'cline' and the 'multicolumn' commands.)

The code which follows is the code for the table shown at the beginning of this chapter and will be discussed in some detail in order to help you understand the details of the tabular commands.

```
1  \begin{table}[here]
2  \begin{center}
3  \begin{threeparttable}
4  \caption{Determination of
5   Hydrogen Spectrum Wavelengths per Order}
6  \label{tab:HSpec}
7  \begin{tabular}{|c | c  c  c|}
8  \hline
9          &       &    Spectral Order   &  \\
10 \cline{2-4}
11  Hydrogen Line & First & Second & Third  \\
12 \hline
13       Red     & 657.0 & 656.7 & 656.3 \\
14       Blue    & 487.3 & 486.7 & 486.1 \\
15       Violet  & 433.5 & 434.3 & ()\tnote{*}\\
16 \hline
17 \end{tabular}
18 \begin{tablenotes}
19 \item[*]Not visible in third order.
20 \end{tablenotes}
21 \end{threeparttable}
22 \end{center}
23 \end{table}
```

The first line[1] of the input text file opens the table environment with the option [here] which forces the (floating) table to be placed at this point in the text (rather than on the next page). The next two lines center the table and open the enhanced threeparttable environment (you must have the `usepackage{threeparttable}` command in the preamble of your document). The fourth and fifth lines define the caption that will be printed with the table, while line six assigns a label to this table for future reference.

Line 7 actually defines the basic layout of the table. The command

```
\begin{tabular}{| c | c  c  c |}
```

opens the tabular environment and tells the compiler that there will be four (4) columns, whose contents will be centered (c), and that there will be vertical lines drawn on the left of column one and on the right of column 4. There will also be a vertical line drawn between column 1 and 2. In lines 8, 12, and 16 the `\hline` command is given to draw horizontal lines (lines 8 and 16 draw the lines at the top and bottom of the tabular region).

Line 9 is the first row of the table. Each column of this row is separated by a column separator (&), and the row ends with the `\\` command sequence. You will notice that most of the columns in this first row have been left empty. The second row of the table is on line 11. Notice that each column of this row contains

[1] Line numbering is available in the 'Verbatim' environment of the 'fancyvrb' package.

data and that each column of data is separated by the column separator (&). The end of this row is again signaled by the \\ symbol. Practically anything can go in the 'cells' of the table, and the contents of the column are *automatically* centered. By placing the text of row one in the third column, I knew that the text I entered would be centered in that column, and leaving column 2 and 4 empty meant that this text would be centered over those three columns. (You can highlight the headings by simply making them bold.)

Now look back at line 10. This line contains a command that tells the LaTeX compiler to draw a horizontal line from column 2 to column 4. If you were to type `cline(3-3)` it would simply draw a horizontal line across column 3.

When making entries in the tabular environment, spaces are generally ignored. This means you can judiciously type the input file in such a way that the column markers (`&`) are vertically aligned and the tabular structure is obvious. This is often helpful when constructing the table of data and also in locating errors in your document. In fact, the curly braces on line 7 containing the layout information can be be placed on a separate line, following the command line. The locations of the column indicators and the vertical lines can then be spread out so that they will line up with the column delimitors and entries that will come below. I could not do that in this document because of limitations of space.

You now have the tools to create a wide variety of tables, and, as seen in the example above, you can also enter 'tablenotes' (footnotes associated with the table) to clarify different items in the table (see line 15, 18–20).

Chapter 5

Including Graphics

To include figures in your document, you will need to include the *graphicx* package at the beginning of your document. You accomplish this with the `\usepackage{graphicx}` command. Within the input document, you will use the associated `\includegraphics` command in the form:

```
\includegraphics[options]\{graphics-filename.ext}
```

This command should be placed inside a *figure* environment, so that the graphic can be centered, and so that a label and caption can be added to the graphic as shown in the example below.

The figure below was produced with the graphing program DrafixCad, which produces ".cad" files that cannot be read by LaTeX. However, you can create a ".pdf" version of this file by printing it to a pdf-printer (you select the pdf-printer from the drop-down menu on the "print" menu screen if one is installed). You can install the PrimoPDF printer driver (it's free!) from the Web. When you select the PrimoPDF printer to print your file, do not select "print to file"; Primo will do that automatically, and will ask where to save the file. By default, PrimoPDF will try to print the file to the desktop. Just change the name of the file to something you will recognize and PrimoPDF will save the image as a ".pdf" file on your desktop. [Note: Avoid using spaces in filenames that you will use with LaTeX. This will help eliminate some unpredictable results.] You do not need to make any changes to the way Primo prints your files; the default settings are what you want to use. (For example, there are advanced settings that allow you to change the output to an ".eps" file—but don't do it – it doesn't work properly!)

Once the ".pdf" file appears on your desktop, move it to the folder containing your ".tex" documents. Then you can use the `\includegraphics` command in the form:

```
\includegraphics[options]\{graphics-file.pdf}
```

to place the image in your document.

The text required to include the "fishbowl" diagram is shown below, along with the printed graphic.

```
\begin{figure}[ht]
\centering
\includegraphics[angle=-90,width=0.80\textwidth]
  {fishbowl.pdf}
\caption{The correct measurement of the length
  of the moving fish requires that you measure
  both ends \emph{at the same time}.}
\end{figure}
```

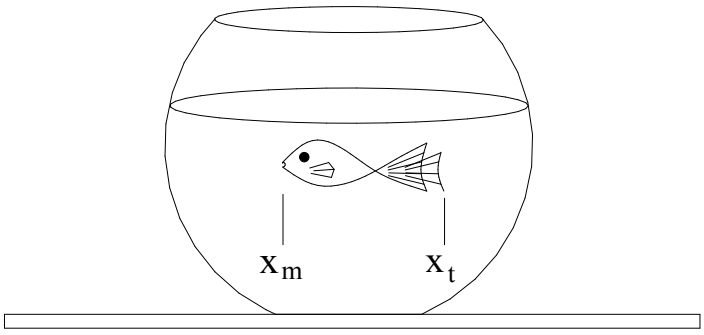

Figure 5.1: The correct measurement of the length of the moving fish requires that you measure both ends *at the same time*.

If you look at the input file for this figure, you will notice the "here" option [ht] entered after the `\begin{figure}` command. This command instructs the LaTeX compiler to place the "floating" figure at this point in the document, not at the bottom of the page, or on some other page. Also the options for the picture indicate that the original image had to be rotated by 90 degrees (counter-clockwise) and that the width of the image was adjusted to 0.80 times the textwidth. The original image was printed as a full page. All scaling and rotating was accomplished by the LaTeX compiler. When you create a graph and print it using the ".pdf" printer, what you often produce is a full page output. Your actual graphic, however, may only take up the central part of the page. If you simply import this file as it is, you will have a considerable amount of "white" space surrounding your graphic. To illiminate this excess white space,

you can use the *viewport* and *clip* options of the `\includegraphics` command as shown in the next example.

```
\begin{figure}[ht]
\centering
\includegraphics[angle=-90,
width=0.8\textwidth,
viewport=1.5in 0in 7in 12in,
clip=true]
{fishbowl.pdf}
\caption{Locating the ends of
 a moving fish in a fish bowl.}
\label{fishbowl}
\end{figure}
```

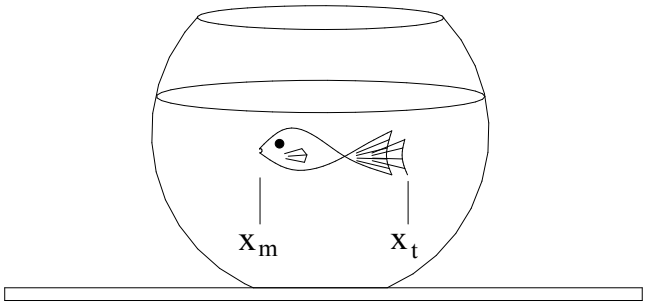

Figure 5.2: Locating the ends of a moving fish in a fish bowl.

As you can see, this has significantly cut down the excess white space at the top and the bottom of the image. To get the correct dimensions, you must examine the *original* drawing. If it is printed in landscape mode, the top, left-hand corner of the page is the origin for LaTeX. Measure the (x_{ll}, y_{ll}) coordinates (in inches or other units) relative to this point for the lower left-hand (ll) corner of the "viewport", and then the (x_{ur}, y_{ur}) coordinates for the upper right-hand (ur) corner of the "viewport" and enter these as

$$\text{viewport} = x_{ll}\, y_{ll}\, x_{ur}\, y_{ur}$$

with just spaces and no commas. You *must* include units with the numbers. And to exclude the part of the image outside the viewport, you must include the "`clip=true`" option (or, you can simply use "`clip`" since the "true" option is the default).

It is advisable to take the time to produce a graph (in .pdf format) that appears as near as possible to the way you want it to appear on the page (without excess white space, etc.), since adjusting the part of a graph and the

size and shape is sometimes quite tedious to get just like you want using scaling and clipping options.

You can include practically anything that can be printed to a .pdf file in your LaTeX document. For example, you can include a graph produced with the Microsoft Excel program by printing the graph with PrimoPDF. Just select the PrimoPDF printer from the printer menu. An example of the LaTeX commands for displaying a graph produced in this way is shown below:

```
\newpage
\begin{landscape}
\begin{figure}[ht]
\centering
\includegraphics[angle=-90,width=1.2\textwidth]
   {Gr_Anal.pdf}
\caption{A graph of distance versus time for the
motion of a falling ball in a gravitational field.}
\end{figure}
\end{landscape}
```

Again the original graph was printed as a full page, and the scaling is accomplished through LaTeX.

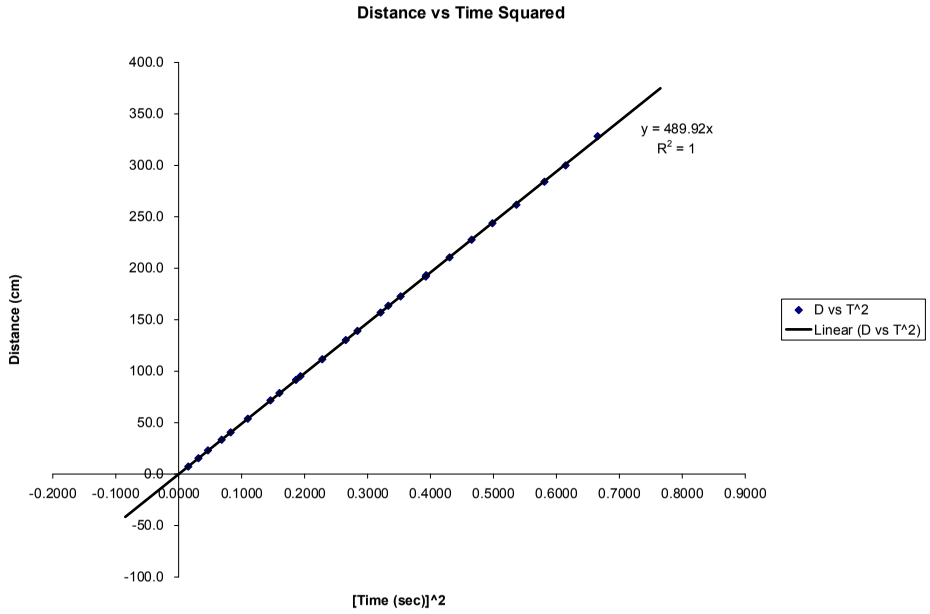

Figure 5.3: A graph of distance versus time for the motion of a falling ball in a gravitational field.