LaTex Guide: Example-Based Approach

Ginger Gengar

8^{th} May 2021

Contents

0.1 Introduction

0.1.1 Rationale

Microsoft Word comes pre-installed in most machines and so most people use Microsoft word. The first thin about LaTex is that this typesetting system is not WYSIWYG (What You See Is What You Get). `LaTex` as a typesetting system separates the content from the actual form of the document, so that users can just focus on what the content of the document should contain first, and then worry about the document's appearance later using a bunch of `macros` or `environments`. I will explain in greater detail what those are. Just the separation between content and appearance is already a good thing as it encourages the user to tackle one problem at a time and not be overwhelmed.

The second aspect as to why `LaTex` is much better than word is that `LaTex` is `cli` based meanwhile word is `gui` based. `cli` here stands for "command line interface" meanwhile `gui` represents "graphical user interface". A `cli` based typesetting system is much faster and convenient to write in compared to a `gui` system, though it might not seem like it at first. To write in `cli` one can just abandon the mouse and type all necessary commands. In `gui` however, one has to navigate to the specific option, scan the entire menu for the type of equation one wants to make and then select it and so forth. Not abandoning the mouse and having to click and search takes time and hinders speed and convenience of writing long mathematical equations.

Another advantage of `LaTex` is that it's beautiful, and neat. The equations in `LaTex` are written so beautifully compared to word. I will give you an example,

$$f(x) = \sum_{n=0}^{\infty} \left[\frac{1}{n!} \frac{d^n f}{dx^n} (x-a)^n \right]$$

Try recreating something as neat as that in word. The equation above is quite standard and easy to produce in LaTex without even any further configurations.

0.1.2 Guide Usage

This guide is meant to be used for the absolute beginner to `LaTex`. The way this guide works is to show by example. The readers are HEAVILY ENCOURAGED to try the examples for themselves to learn how the `LaTex` typsetting system really works. This guide is meant to "teach by inferrence" wherein snippets of code is shown and then based on the results they produce, the readers can "infer" the rules of the `LaTex` typesetting system.

0.1.3 Computing Environment

`LaTex` as a typesetting system works absolutely well with a standard `Linux` operating system, but also works fine for `Windows` system. Before I had migrated to `Linux` I used MikTex on windows. There are other `LaTex` distributions out there that can work. After installing `LaTex` using the package manager in `Linux` or through `MikTex` or some other distribution in windows, it is important to keep in mind that `LaTex` is compiled. Here are the steps to writing in `LaTex`:

1. Write code in `LaTex` using a text editor: On a linux system, one can choose their own text editors, such as `vim`, `emacs`, `nano`, `sublime`, just to name a few. These text editors work the same. All we are doing by invoking the text editor is writing some `ASCII` words into a file in a directory of our choosing.

2. Compile the `LaTex` code using a compiler (mine is `pdflatex`): The compilation process here typically uses the terminal. To compile a `.tex` file named `myfile.tex` using `pdflatex`, do this `pdflatex myfile.tex`.

3. Examine the generated PDF document, or, if an error occured, examine the erorr, fix the LaTex code and retry compiling at step 2.

When I first started I made alot of errors. If we make errors in not having balanced braces or invoked an invalid command, then the LaTex compiler will complain and no document is generated, because code is invalid. When I first started it was so difficult for me to interpret the common error messages. So it is advised to compile the document often, so that if an error were to occur, there is only a few changes before last compilation and it would help significantly in debugging LaTex code.

Of course, this is the common beginner experience. As you use LaTex and gain more familiarity with it, you will naturally avoid all the common errors and writing LaTex rarely invokes any new errors, thus making LaTex a sane and effective means to communicate formal ideas.

0.2 The Simplest Document

The simplest LaTex document that I can produce is shown below. The document below prints only "This is some text".

```
\documentclass[a4paper, 12pt]{report}

\begin{document}

This is some text.

\end{document}
```

To me, it seems that only 3 things are needed for the absolute minimum of a LaTex document:

1. Declaration at the top of what "Type" of Document one wants to generate: `documentclass[a4paper, 12pt]{report}`
2. Environment block statements: `begin{document}` followed by a `end{document}`
3. Some small content inside the Environment block statement: `This is some text.`

Earlier I mentioned "macros" and "environments". An environment was invoked in the LaTex document above. The LaTex document above contains a single environment: `document`.

0.3 Commenting Lines

The base LaTex typesetting system acknowledges that a line beginning with % contains comments. All lines beginning with a % would be ignored by the LaTex compiler. An example is shown below,

```
\documentclass[a4paper, 12pt]{report}

\begin{document}

This is some text.
%This is just a comment. LaTex Will not print anything on this line

\end{document}
```

You can make comments using the % anywhere in a LaTex document. The example below is a valid LaTex document and demonstrates this fact:

```
%You can even put comments here
\documentclass[a4paper, 12pt]{report}

%Another Comment was declared here
```

```
\begin{document}

This is some text.
%This is just a comment. LaTex Will not print anything on this line

\end{document}
%Or Here, It doesn't matter where you put comments
```

Comments are useful for encoding messages that are just for the author and not for the reader's to see. For example, when writing a long document, perhaps some part of the document needs to be made clearer or is wrong. It would be useful to add comments to mark where changes are necessary. The method shown above works great, except that it is limited to single line comments.

0.4 Introduction to Packages

The most simple document we studied earlier is shown below,

```
\documentclass[a4paper, 12pt]{report}

\begin{document}

This is some text.

\end{document}
```

Comments were added to mark the "Preamble" region and the "Document Body" region.

```
\documentclass[a4paper, 12pt]{report}

%This is the place between the document class definition
%and the "start" of the document by the command \begin{document}
%This particular region is called the pre-amble

\begin{document}

%This is is the region between the \begin{document}
%and the \end{document}. This is the "body" of the document

This is some text.

\end{document}
```

The preamble is where packages are included for the rest of the document. Packages extends the commands a programmer can use, thereby extending `LaTex` functionalities. Earlier, I mentioned how it is only possible to comment single lines with the `%` command. To make multi-line commented blocks, we need to use the `comment` package in order to declare multi-line commented blocks. An example of how to load the package is shown below,

```
\documentclass[a4paper, 12pt]{report}

%This is the pre-amble region

%This is the command to load the comment package
\usepackage{comment}

\begin{document}

%This is the document body

This is some text.

\end{document}
```

Note in the example above, all we have done is load the comment package. We have not used the comment package yet. The comment package adds a bunch of utilities, but most importantly it declares a new environment: comment. Environments were introduced briefly and previously. Below shows a demonstration of how to use the comment environment in making multi-line block comments:

```
\documentclass[a4paper, 12pt]{report}

%This is the pre-amble region

%This is the command to load the comment package
\usepackage{comment}

\begin{comment}
This is a comment block
More lines can be added
and the LaTex Compiler would ignore
All of the lines we put in here
This is some more lines
and more lines
and more lines
\end{comment}

\begin{document}

%This is the document body

This is some text.

\begin{comment}
This is a comment block
More lines can be added
and the LaTex Compiler would ignore
All of the lines we put in here
This is some more lines
and more lines
and more lines
\end{comment}

\end{document}
```

The statement usepackage{comment} is necessary. Without this, the LaTex document would not compile. Try commenting that statement out and try compiling and see what happens. The comment environment must be used after the usepackage statement. This is because using the LaTex compilers read the documents sequentially and would not recognize the environment before it is loaded.

0.5 Document Margins

The document below is just a simple document that contains some text. The text shown below is long enough to fit easily into multiple lines.

```
\documentclass[a4paper, 12pt]{report}

\begin{document}

Flow separation is one of the most important things to avoid in the design of aircraft wings. Flow
    separation is accompanied by dramatic increase in drag and loss of lift. When an airfoil is tested
     in the transonic regime, the flow over the airfoil goes supersonic on the front of the airfoil.
    In the pressure recovery phase where the flow slows down to subsonic speeds, a shock wave might
    occur. This shockwave impinges on the airfoil and typically separation follows at this point.
    Airfoils which are not designed at transonic speeds would experience poor performance at transonic
     flight speeds.

\end{document}
```

Try compiling it. The document as is kind of looks "stifled" the margins are kind of too wide and it would be great if the margins could be made thinner. This is exactly what the `geometry` package allows us to do. Consider an addition of a few lines in the pre-amble as shown below,

```
\documentclass[a4paper, 12pt]{report}

\usepackage{geometry}

\geometry{portrait, margin= 0.8in}

\begin{document}

Flow separation is one of the most important things to avoid in the design of aircraft wings. Flow
    separation is accompanied by dramatic increase in drag and loss of lift. When an airfoil is tested
     in the transonic regime, the flow over the airfoil goes supersonic on the front of the airfoil.
    In the pressure recovery phase where the flow slows down to subsonic speeds, a shock wave might
    occur. This shockwave impinges on the airfoil and typically separation follows at this point.
    Airfoils which are not designed at transonic speeds would experience poor performance at transonic
     flight speeds.

\end{document}
```

The statement `usepackage{geometry}` loads the geometry package and the statement `geometry{portrait, margin= 0.8in}` determines what the size of the margins are going to be and also the layout of the document. Try changing the margin value and the configuration to `landscape` from `portrait`.

0.6 Basic Equations

Consider the expression $x^2 = 3$. Let us try to write that in `LaTex`. Begin by writing this inside the body of a `LaTex` document:

```
\begin{equation}
x^{2} = 3
\end{equation}
```

The output of the code above is shown below.

$$x^2 = 3 \tag{1}$$

Here we are invoking a basic `LaTex` environment `equation`. There is a primitive command in `Tex` for making simple equations,

```
$$x^{2} = 3$$
```

The output of the primitive `LaTex` code is shown below,

$$x^2 = 3$$

This has certain issues possibly with inconsisten vertical spacing, so it is recommended to do this instead,

```
\[x^{2} = 3\]
```

The output of the `LaTex` code is shown below,

$$x^2 = 3$$

It really depends on personal preference, but I've used the `$$ $$` command and it has worked so far. Another important thing to note is that spaces do not matter in a `LaTex equation` environment. Here is an example,

```
$$x = ab$$
```

The output of the LaTex code of the product of 2 numbers without space is shown below,

$$x = ab$$

Now adding alot of space between a and b,

```
$$x = a            b$$
```

The result of the code above is shown below,

$$x = ab$$

This shows that spacing in the variables do not matter in the equation environment or has no effect on how the equation is interpreted by the LaTex compiler.

0.7 Amsmath Package

The amsmath package is massive package that extends the mathematical typesetting capabilities of LaTex. The amsmath package can be loaded using the command

```
\usepackage{amsmath}
```

amsmath is such an important package that almost any LaTex document that has any substantial mathematical expressions would need to use amsmath.

0.7.1 Suppressing Equation Labels

The amsmath package overwrite some of the basics of the equation environment. Adding a * symbol on the equation environment suppresses the labellings. The example below would demonstrate this:

```
\begin{equation*}
x^{2} = 3
\end{equation*}
```

The output of the code above is shown below,

$$x^2 = 3$$

Why would we ever want to suppress the labelling? Imagine you were writing a long mathematical derivation, and you wrote every single equation with labelling unsuppressed. The last line would be labelled with some obscenely high number due to the sheer amount of steps that was done in the derivation. So it is unruly to see meaningless labels and is best to suppress the labels except for the time the equations need to be referred back to.

0.7.2 Aligning Equations

0.7.3 Multi-Equation Lines

Imagine that you want to communicate a bunch of simple equations. Putting the each equation on a newline, is the default, but that would leave alot of wasted space. Here is an example of what I mean:

```
$$x = k$$
$$y = q$$
$$z = u$$
$$b = m$$
```

The outuput of the LaTex code above is shown below,

$$x = k$$

$$y = q$$

$$z = u$$

$$b = m$$

It would be far more compact to write those simple equation in a single line. A naive approach of that would be to do this:

```
$$x = k  y = q  z = u  b = m$$
```

The output of the naive approach is shown below,

$$x = ky = qz = ub = m$$

The equations look completely messed up, because there are no gaps between them. The `amsmath` package provides a way to add gaps between equations, using the command `\quad`. A more reasonable approach is to add `\quad` in between the equations, like this:

```
$$x = k \quad y = q \quad z = u \quad b = m$$
```

The output of the `LaTex` code with `\quad` is shown below,

$$x = k \quad y = q \quad z = u \quad b = m$$

The result looks okay, but it could be made better by adding a comma in between to separate the equations,

```
$$x = k \quad, y = q \quad, z = u \quad, b = m$$
```

The output of the `LaTex` equation above is shown below,

$$x = k \quad , y = q \quad , z = u \quad , b = m$$

My personal preference looks something like this instead,

```
$$x = k \quad,\quad y = q \quad,\quad z = u \quad,\quad b = m$$
```

$$x = k \quad , \quad y = q \quad , \quad z = u \quad , \quad b = m$$

0.8 Macros

Macros are a set of commands that allow the programmer to redefine typical `LaTex` command as something else. These commands are typically declared in the pre-amble of a document.

0.8.1 Usage & Syntax

Consider the following command declared in the pre-amble of a document:

```
\def\t{\theta}
```

The command `\def` is a special keyword, which allows the command inside the brackets, in this case `\theta` to be redefined as the command in front of the bracket, which is `\t`. Therefore, when the command shown below is invoked.

```
$$\t$$
```

The output of the `LaTex` code shown above,

$$\theta$$

Here is a list of Macros I had personally used in the past,

```
\def\t{\theta}
\def\a{\alpha}
\def\be{\beta}
\def\w{\omega}
\def\la{\lambda}
\def\g{\gamma}
\def\f{\frac}
\def\l{\left}
\def\r{\right}
\def\dst{\displaystyle}
\def\b{\bar}
\def\h{\hat}
\def\ph{\phi}
\def\d{\cdot}
\def\n{\nabla}
\def\p{\partial}
\def\lap{\mathcal{L}}
```

With the macros defined as such, code as shown below,

```
$$\t \quad,\quad \a \quad,\quad \ph \quad,\quad \lap$$
```

compiles into an equation shown below,

$$\theta \quad , \quad \alpha \quad , \quad \phi \quad , \quad \mathcal{L}$$

0.8.2 Advantages and Disadvantages

Should anyone use Macros? Macros are one way to write in LaTex faster. By redefining an otherwise long command such as \frac{}{} one can simply write \f{}{}, which improves speed of writing a LaTex document. This is not too obvious since it only saves 3 keystrokes, but what about \left as \l and \right as \r? Those macros save 4 keystrokes for right and 3 keystrokes for left. The small increments in ease stacks up when writing increasingly long documents.

That being said, I believe that macros are evil. The issue with macros is that the commands we redefine is author specific. If Sue redefines \displaystyle as \dst, everytime Mary looks at \displaystyle she will get confused because \displaystyle is not part of the standard LaTex command. If Sue and Mary work together, then Mary would have to keep referring to the preamble to translate the Macros, or Mary and Sue must work in completely different parts of the document and cannot easily check each other's work. This is just for a team of 2. But what happens if a project is sufficiently large and has 7 people working on it? Does it mean we need 7 different Macros redefining \frac{}{}? It is silly and that is why macros should be avoided.

So is there another way? Yes. There are cleaner ways to reduce keystrokes while typing. The way to reduce keystrokes is by using snippets. snippets allow a text editor such as vim to expand a string into another string, and snippets are configured by the author. For example, in my snippet setup, dst will automatically expand to \displaystyle. Therefore, I can enjoy the same benefits of reduced keystrokes while writing, while retaining readability by using just the standard LaTex commands.

There are other ways as well that make Macros obselete. For example, vim has a plugin named vimtex which allows the adding of the \left and \right commands to a set of brackets by just keying tsd, which allows for even more effective writing.

0.9 Common Mathematical Equations

Now the "fun parts" of mathematical writing in LaTex begins.

0.9.1 Greek Symbols

A list of the greek symbols that are available using the base LaTex is shown below. Some of the lowercase greek symbols shown below do not have an uppercase equivalent. This is because the uppercase equivalent is just a typical latin alphabet capital. For example. \alpha is valid, but \Alpha is not valid. It seems that if one wants to represent \Alpha one can just invoke A, which results in A.

Command	Symbol	Command	Symbol	Command	Symbol	Command	Symbol
\alpha	α	\beta	β	\gamma	γ	\Gamma	Γ
\delta	δ	\Delta	Δ	\epsilon	ϵ	\zeta	ζ
\eta	η	\theta	θ	\Theta	Θ	\iota	ι
\kappa	κ	\lambda	λ	\Lambda	Λ	\mu	μ
\nu	ν	\xi	ξ	\Xi	Ξ	\pi	π
\Pi	Π	\rho	ρ	\sigma	σ	\Sigma	Σ
\tau	τ	\upsilon	υ	\Upsilon	Υ	\phi	ϕ
\Phi	Φ	\chi	χ	\psi	ψ	\Psi	Ψ
\omega	ω	\Omega	Ω	None	None	None	None

0.9.2 Subscripts & Superscripts

Often times, it is important to make a distinction between variables. The subscript is used to make such a distinction. This is an example of how to use a subscript. LaTex should print x subscripted by a.

```
$$x_{a}$$
```

The output of the LaTex code above,

$$x_a$$

Earlier, I have already shown how to do a superscript, though I did not formally declare how to use superscript. The example below is going to print x superscripted by a.

```
$$x^{a}$$
```

The output of the superscript is shown below,

$$x^a$$

The subscripts and superscripts accept whatever is inside the brackets. Below is an example that demonstrates putting whatever we want inside the brackets,

```
$$x^{abc/d} \quad,\quad x_{abc/d}$$
```

Please do not be intimidated by the \quad,\quad part. This was explained in section 0.7.3. This command only adds a space followed by a , and then another space. The output of the LaTex code above is shown below,

$$x^{abc/d} \quad , \quad x_{abc/d}$$

The subscripts and superscripts can be used at the same time in the same equation. The LaTex code below will show some a polynomial expression demonstrating how subscripts and superscripts can be used at the same time,

```
$$0 = k_{a}^{2}x_{i}^{3} + m_{s}^{3}x_{i}^{2} + l_{r}^{2}x_{i} + k_{b}$$
```

The output of the LaTex code is shown below,

$$0 = k_a^2 x_i^3 + m_s^3 x_i^2 + l_r^2 x_i + k_b$$

Note that when we the order at which a subscript and superscript is written for a variable does not matter. For example,

```
$$k_{a}^{2} \quad,\quad k^{2}_{a}$$
```

The output of the LaTex code is shown below,

$$k_a^2 \quad , \quad k_a^2$$

Subscripts and superscripts might be enough for most applications, but what if we wanted even more control of how a variable is specified? What if we wanted to add small subscripts and superscripts at the front of a variable? This might sound like a silly idea after all, it is a rarity to find any mathematical working with such an odd form of notation. I have encountered this problem before in particular in the fields of dynamics, wherein it was necessary to specify alot of information about a variable such as its basis vectors, the reference frame it is nested in, the object the quantity belongs to, and a counter variable. Consider the following LaTex code:

```
$${}^{a}_{b}$$
```

The output of the strange LaTex code is shown below,

$${}^{a}_{b}$$

So the LaTex code above subscripts an empty set by b and superscripts the same empty set by a. Since the empty set represented by {} does not represent anything, the compiler prints a on top of b. So the code above can be used to give forward subscript and superscript to some variable v. The code below demonstrates this,

```
$${}^{a}_{b}v$$
```

The LaTex code prints,

$${}^{a}_{b}v$$

From then on, it is simple enough to add the conventional superscripts and subscripts to the variable to give the variable v a combined total of 4 specifiers, a, b, c and d :

```
$${}^{a}_{b}v^{c}_{d}$$
```

The LaTex code above produces the result below,

$${}^{a}_{b}v^{c}_{d}$$

0.9.3 Fractions & Brackets

Usually there are 3 levels of conventional mathematical brackets, those are demonstrated below,

$$\{[()]\}$$

The most common bracket is generated by the code shown below,

```
$$()$$
```

The output of the LaTex code is shown below,

$$()$$

The 2^{nd} level of bracket is generated by the code below,

```
$$[]$$
```

The output of the LaTex code is shown below,

$$[]$$

The outermost bracket is generated by the code below,

```
$$\{\}$$
```

The output of the LaTex code is shown below,

$$\{\}$$

The last bracket is kind of unique. The issue with just commanding {} is that the characters { and } are special characters in the LaTex typesetting system. Therefore, \must be used in front to specify the bracket characters as truly brackets and not special LaTex characters. The way to write fractions in LaTex is to invoke the \frac{}{} command. The 1^{st} set of brackets would be the numerator meanwhile the 2^{nd} set of brackets would be the denominator. An example of that is shown below,

```
$$\frac{numerator}{denominator}$$
```

The resulting LaTex output is shown below,

$$\frac{numerator}{denominator}$$

Now it is quite typical to encounter a fraction nested within another fraction. In default LaTex, a situation like this would make the nested fraction usually smaller. An example of that is shown below,

```
$$f = \frac{1}{1+\frac{3}{A}}$$
```

The output of the LaTex code is shown below,

$$f = \frac{1}{1+\frac{3}{A}}$$

Sometimes this is acceptable or desirable, but personally I find this ugly. So, to inflate the nested fraction to its original size, we can invoke the \displaystyle command. Here is an example of the \displaystyle command,

```
$$f = \frac{1}{\displaystyle 1+\frac{3}{A}}$$
```

The corrected LaTex code output is shown below,

$$f = \frac{1}{\displaystyle 1+\frac{3}{A}}$$

Here, notice how the \displaystyle command has returned the nested fraction to its original size. Do note that the \displaystyle command needs to be declared in the region of the nested fraction, which in this case is the numerator of the "main" fraction. Often times, it is necessary to add brackets to fractions. Consider the normal fraction,

```
$$x = \frac{1}{2}$$
```

The output of the LaTex code above is shown below,

$$x = \frac{1}{2}$$

What if we naively declared brackets on the fraction like below:

```
$$x = (\frac{1}{2})$$
```

The output of the LaTex code above is shown below,

$$x = (\frac{1}{2})$$

The brackets do not span the entire height of the fraction and this looks very ugly. So we can fix this by invoking the \left and \right on the brackets like this,

```
$$x = \left(\frac{1}{2}\right)$$
```

The output of the LaTex code above is shown below,

$$x = \left(\frac{1}{2}\right)$$

The commands \left and \right is also compatible for other types of brackets as well, here is an example:

```
$$x = \left[\frac{1}{2}\right]$$
```

The output of the LaTex code above is shown below,

$$x = \left[\frac{1}{2}\right]$$

```
$$x = \left\{\frac{1}{2}\right\}$$
```

The output of the LaTex code above is shown below,

$$x = \left\{\frac{1}{2}\right\}$$

the \left and \right command combined together with the previous \displaystyle example:

```
$$f = \left\{\left[\left(\frac{1}{\displaystyle 1+\frac{3}{A}}\right)\right]\right\}$$
```

The output of the LaTex code above is shown below,

$$f = \left\{\left[\left(\frac{1}{\displaystyle 1+\frac{3}{A}}\right)\right]\right\}$$

0.9.4 Basic Calculus Operations

Derivatives

The code below demonstrates how to show a derivative operator,

```
$$\frac{d}{dx}$$
```

The LaTex code above, produces the output shown below,

$$\frac{d}{dx}$$

Note that a simple derivative operator is just a fraction whose numerator is d and denominator is d var wherein you can chooose var the name of a variable of your choosing. To demonstrate the different choices of variables,

```
$$\frac{d}{dx} \quad,\quad \frac{d}{dy} \quad,\quad \frac{d}{dz} \quad,\quad \frac{d}{d \alpha} \quad
    ,\quad \frac{d}{d \beta} \quad,\quad \frac{d}{d \gamma} \quad,\quad \frac{d}{d \xi}$$
```

The LaTex code above, produces the output shown below,

$$\frac{d}{dx} \quad , \quad \frac{d}{dy} \quad , \quad \frac{d}{dz} \quad , \quad \frac{d}{d\alpha} \quad , \quad \frac{d}{d\beta} \quad , \quad \frac{d}{d\gamma} \quad , \quad \frac{d}{d\xi}$$

Sometimes we want to take derivative of a variable or function by "inlining" it to the derivative operator, here is an example of how to do that,

```
$$\frac{d f(x)}{dx}$$
```

The LaTex code above, produces the output shown below,

$$\frac{df(x)}{dx}$$

By the syntax of \frac{}{}, it is sufficient to just add the object that is differentiated to the numerator of the derivative fraction. Preferrably, we might want to separate the derivative operator and the object of derivative. This shows how,

```
$$\frac{d}{dx}[f(x)]$$
```

The LaTex code above, produces the output shown below,

$$\frac{d}{dx}[f(x)]$$

Taking multiple derivatives consecutively can be done by a simple subscript to the derivative numerators and denominators like so,

```
$$\frac{d^{2}y}{dx^{2}} \quad,\quad \frac{d^{3}y}{dx^{3}} \quad,\quad \frac{d^{4}y}{dx^{4}} \quad,\
    quad \frac{d^{n}y}{dx^{n}}$$
```

The LaTex code above, produces the output shown below,

$$\frac{d^{2}y}{dx^{2}} \quad,\quad \frac{d^{3}y}{dx^{3}} \quad,\quad \frac{d^{4}y}{dx^{4}} \quad,\quad \frac{d^{n}y}{dx^{n}}$$

Preferrably if one seeks to separate operators,

```
$$\frac{d^{2}}{dx^{2}}[f(x)] \quad,\quad \frac{d^{3}}{dx^{3}}[f(x)] \quad,\quad \frac{d^{4}}{dx^{4}}[
    f(x)] \quad,\quad \frac{d^{n}}{dx^{n}}[f(x)]$$
```

The LaTex code above, produces the output shown below,

$$\frac{d^{2}}{dx^{2}}[f(x)] \quad,\quad \frac{d^{3}}{dx^{3}}[f(x)] \quad,\quad \frac{d^{4}}{dx^{4}}[f(x)] \quad,\quad \frac{d^{n}}{dx^{n}}[f(x)]$$

Suppose we had a fraction in object we would like to differentiate, the results of section 0.9.3 would allow the brackets to span the entire fraction. Below is a demonstration

```
$$\frac{d^{n}}{dx^{n}}[\frac{f(x)}{g(x)}] \quad,\quad \frac{d^{n}}{dx^{n}}\left[\frac{f(x)}{g(x)}\
    right]$$
```

The LaTex code above, produces the output shown below,

$$\frac{d^{n}}{dx^{n}}[\frac{f(x)}{g(x)}] \quad,\quad \frac{d^{n}}{dx^{n}}\left[\frac{f(x)}{g(x)}\right]$$

The one on the left is without the \left and \right commands. The brackets are stunted. THe right invokes the \left and \right command, the brackets are appropriately sized. So the findings from the previous sections can be reused in conjunction with each other to form increasingly complicated expressions with simple commands. It is also possible to change the total derivative with the partial derivative. Instead of using d use the command \partial. Below demonstrates how to do partial derivatives instead of normal derivatives,

```
$$\frac{\partial^{2}}{\partial x^{2}}[f(x)] \quad,\quad \frac{\partial^{3}}{\partial x^{3}}[f(x)] \
    quad,\quad \frac{\partial^{4}}{\partial x^{4}}[f(x)] \quad,\quad \frac{\partial^{n}}{\partial x^{n
    }}[f(x)]$$
```

The LaTex code above, produces the output shown below,

$$\frac{\partial^{2}}{\partial x^{2}}[f(x)] \quad,\quad \frac{\partial^{3}}{\partial x^{3}}[f(x)] \quad,\quad \frac{\partial^{4}}{\partial x^{4}}[f(x)] \quad,\quad \frac{\partial^{n}}{\partial x^{n}}[f(x)]$$

Integrals

The integral symbol could be invoked using the `\int` command. Below is a demonstration

```
$$\int$$
```

The LaTex code above, produces the output shown below,

$$\int$$

Suppose we wanted to take a simple indefinite integral of a monomial x^2, then we can use the `\int` command shown previously, followed by the monomial, and then ended by `d var` wherein `var` is the variable we are integrating with respect to. Below shows a demonstration of this,

```
$$\int x^{2}dx$$
```

The LaTex code above, produces the output shown below,

$$\int x^2 dx$$

The integral looks decent, except that the integrand which is x^2 in this case, has no gap with the end of the integral, dx. To remedy this, we can invoke the base LaTex command `\,` to add a gap between the integrand and the end of the integral like so,

```
$$\int x^{2}\,dx$$
```

The LaTex code above, produces the output shown below,

$$\int x^2 \, dx$$

This looks much better. After the indefinite integral, how to write a definite integral with bounds? To do that, we can just superscript and subscript the `\int` command to fill the bounds of the integral. Below is a demonstration of this superscripting and subscripting,

```
$$\int^{a}_{b} x^{2}\,dx$$
```

The LaTex code above, produces the output shown below,

$$\int_{b}^{a} x^2 \, dx$$

Of course as with all things in LaTex the integral commands can be combined together to form more complex expressions such as the one below,

```
$$\int^{e}_{f} \int^{c}_{d} \int^{a}_{b} g(x,y,z) \,dxdydz$$
```

The LaTex code above, produces the output shown below,

$$\int_{f}^{e} \int_{d}^{c} \int_{b}^{a} g(x,y,z) \, dxdydz$$

Here is another example, this time for cylindrical coordinates, just to demonstrate the LaTex syntax,

```
$$\int^{h}_{0}\int^{2\pi}_{0}\int^{r_{2}}_{r_{1}} f(r,\theta,z) \,rdr\,d\theta\,dz$$
```

The LaTex code above, produces the output shown below,

$$\int_{0}^{h} \int_{0}^{2\pi} \int_{r_1}^{r_2} f(r,\theta,z) \, r dr \, d\theta \, dz$$

Summation Operations

Earlier I showed how to write the greek symbol \Sigma, which compiles to Σ. This is tiny so don't do this. The true summation operation is invoked using the \sum command. Here is a demonstration,

```
$$\sum$$
```

The LaTex code above, produces the output shown below,

$$\sum$$

The summation operation is just a symbol, similar to the way the integral symbol works. To do say summation of some coefficient a_i:

```
$$\sum(a_{i})$$
```

The LaTex code above, produces the output shown below,

$$\sum(a_i)$$

Other types of brackets can be used as well,

```
$$\sum[a_{i}]$$
```

The LaTex code above, produces the output shown below,

$$\sum[a_i]$$

If the object to be summed over is a fraction, then the commands \left and \right would both be valid as well. Here is an example,

```
$$\sum\left[\frac{a_{i}}{i!}\right]$$
```

The LaTex code above, produces the output shown below,

$$\sum\left[\frac{a_i}{i!}\right]$$

This is fine for more basic equations, but we can add the lower bounds and upper bounds of the summation just like in the case with the integral. Here is an example,

```
$$\sum^{n}_{i = 0}\left[\frac{a_{i}}{i!}\right]$$
```

The LaTex code above, produces the output shown below,

$$\sum^{n}_{i = 0}\left[\frac{a_i}{i!}\right]$$

Let us try reconstructing the taylor series shown earlier in section 0.1.1,

$$f(x) = \sum^{\infty}_{n=0}\left[\frac{1}{n!}\frac{d^nf}{dx^n}(x-a)^n\right]$$

From section 0.9.2, the polynomial part of the taylor series can be written,

```
$$(x-a)^{n}$$
```

The LaTex code above, produces the output shown below,

$$(x-a)^n$$

From section 0.9.4, we can easily construct the first order derivative of function f using fractions and d,

```
$$\frac{df}{dx}$$
```

The LaTex code above, produces the output shown below,

$$\frac{df}{dx}$$

To take n successive derivatives, just use the superscript notations,

```
$$\frac{d^{n}f}{dx^{n}}$$
```

The LaTex code above, produces the output shown below,

$$\frac{d^{n}f}{dx^{n}}$$

Putting these 2 together forms,

```
$$\frac{d^{n}f}{dx^{n}}(x-a)^{n}$$
```

The LaTex code above, produces the output shown below,

$$\frac{d^{n}f}{dx^{n}}(x-a)^{n}$$

Just adding a simple fraction at the beginning with a numerator `1` and denominator `n!` yields,

```
$$\frac{1}{n!}\frac{d^{n}f}{dx^{n}}(x-a)^{n}$$
```

The LaTex code above, produces the output shown below,

$$\frac{1}{n!}\frac{d^{n}f}{dx^{n}}(x-a)^{n}$$

There's only 1 additional new symbol that one has to take care of, the infinity symbol. Here is the syntax for the infinity symbol,

```
$$\infty$$
```

The LaTex code above, produces the output shown below,

$$\infty$$

Putting all that inside a summation operation,

```
$$f(x) = \sum^{\infty}_{n = 0}\left[\frac{1}{n!}\frac{d^{n}f}{dx^{n}}(x-a)^{n}\right]$$
```

The LaTex code above, produces the output shown below,

$$f(x) = \sum^{\infty}_{n = 0}\left[\frac{1}{n!}\frac{d^{n}f}{dx^{n}}(x-a)^{n}\right]$$

This is the taylor expression shown in section 0.1.1.

Product Operations

The product operators are similar to the integral operators and the summation operator. Here is an example,

```
$$\prod^{n}_{i = 0}[a_{i}]$$
```

The LaTex code above, produces the output shown below,

$$\prod^{n}_{i = 0}[a_{i}]$$

The same rules for summation operations also apply for product operations. To find the successive multiplications of fraction objects, just do the same thing with the summation case,

```
$$\prod^{n}_{i = 0}\left[\frac{a_{i}}{i!}\right]$$
```

The LaTex code above, produces the output shown below,

$$\prod^{n}_{i = 0}\left[\frac{a_{i}}{i!}\right]$$

0.9.5 Vector Calculus Notations

The bar vector is typically used to denote that a vector. The way the bar is used is shown below,

```
$$\bar{v}$$
```

The LaTex code above, produces the output shown below,

$$\bar{v}$$

A bar would be drawn over the entirety of whatever is in between { and }. Here is an example of using multiple symbols,

```
$$\bar{vb}$$
```

The LaTex code above, produces the output shown below,

$$\bar{vb}$$

The hat is usually used to denote unit vectors. Its usage and syntax is similar to the bar case,

```
$$\hat{v}$$
```

The LaTex code above, produces the output shown below,

$$\hat{v}$$

For multiple symbols,

```
$$\hat{vb}$$
```

The LaTex code above, produces the output shown below,

$$\hat{vb}$$

Of course, in LaTex, anything can be mixed with each other. So, one can use a subscript and a bar or hat to represent some vector like this,

$$\bar{v}_i \quad , \quad \hat{v}_i$$

When discussing vector calculus, one cannot leave out the operations. Most of the operations are essentially symbols. To do a cross product notation, use \times as shown below,

```
$$\bar{a} \times \bar{b}$$
```

The LaTex code above, produces the output shown below,

$$\bar{a} \times \bar{b}$$

To do a dot product notation, use \cdot as shown below,

```
$$\bar{a} \cdot \bar{b}$$
```

The LaTex code above, produces the output shown below,

$$\bar{a} \cdot \bar{b}$$

To use the gradient operator, invoke the command \nabla as shown below

```
$$\nabla \bar{v}$$
```

The LaTex code above, produces the output shown below,

$$\nabla \bar{v}$$

Since commands can be mixed in LaTex it is also possible to get the curl and divergence of a vector field by using a combination of the operators above,

```
$$\nabla \times \bar{v} \quad,\quad \nabla \cdot \bar{v}$$
```

The LaTex code above, produces the output shown below,

$$\nabla \times \bar{v} \quad , \quad \nabla \cdot \bar{v}$$

0.9.6 Matrices

Matrices in LaTex are characterized by the type of brackets that they have. The pmatrix environment declared matrices with rounded brackets meanwhile the bmatrix environment declares matrices with sharp brackets. There are other different types of matrices other than pmatrix and bmatrix, but for now these 2 should suffice. To write the next horizontal element of a matrix one use &. Here is a simple example,

```
$$\begin{bmatrix}
v_{1} & v_{2} & v_{3}
\end{bmatrix}$$
```

The LaTex code above, produces the output shown below,

$$\begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$$

To have a larger gap between the columns of the matrix, one can use && instead of just &. Here is an example,

```
$$\begin{bmatrix}
v_{1} && v_{2} && v_{3}
\end{bmatrix}$$
```

The LaTex code above, produces the output shown below,

$$\begin{bmatrix} v_1 && v_2 && v_3 \end{bmatrix}$$

To write the next vertical element of a matrix, one can use \\. Here is a simple example,

```
$$\begin{bmatrix}
v_{1} \\ v_{2} \\ v_{3}
\end{bmatrix}$$
```

The LaTex code above, produces the output shown below,

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

To make the rows have a larger separation between them, one can use \\~\\ instead of just \\. Here is an example,

$$\begin{bmatrix} v_1 \\~\\ v_2 \\~\\ v_3 \end{bmatrix}$$

Therefore, putting all that together allows us to write a simple matrix as shown below,

```
$$\begin{bmatrix}
a && b \\
c && d
\end{bmatrix}$$
```

The LaTex code above, produces the output shown below,

$$\begin{bmatrix} a && b \\ c && d \end{bmatrix}$$

In principle, this matrix can be of any size. Therefore, a 3×3 matrix could be written below,

```
$$\begin{bmatrix}
v_{11} && v_{12} && v_{13} \\
v_{21} && v_{22} && v_{23} \\
v_{31} && v_{32} && v_{33} \\
\end{bmatrix}$$
```

The LaTex code above, produces the output shown below,

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix}$$

We could also leave some entries blank and nothing will be printed for those elements. Here is an example to demonstrate,

```
$$\begin{bmatrix}
v_{11} && v_{12} && v_{13} \\
v_{21} &&  && v_{23} \\
v_{31} && v_{32} && v_{33} \\
\end{bmatrix}$$
```

The LaTex code above, produces the output shown below,

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix}$$

LaTex matrices can also be superscripted and subscripted. Here is an example of that,

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}_{x}^{3}$$

A common mathematical statement regarding transposes therefore can already be written with the tools covered,

```
$$\begin{bmatrix}
v_{1} && v_{2} && v_{3}
\end{bmatrix}^{T} = \begin{bmatrix}
v_{1} \\ v_{2} \\ v_{3}
\end{bmatrix}$$
```

The LaTex code above, produces the output shown below,

$$\begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^{T} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

It is sometimes needed to speak about general-sized matrices, at which dots would be useful. To use horizontal dots in a matrix, one can just use the \dots symbol as one of the matrix elements. Here is an example of this,

```
$$\begin{bmatrix}
v_{1} & \dots & v_{n}
\end{bmatrix}$$
```

The LaTex code above, produces the output shown below,

$$\begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$$

to invoke vertical dots this time, one can use the \vdots symbol as one of the matrix elements, such as shown below,

```
$$\begin{bmatrix}
v_{1} \\ \vdots \\ v_{n}
\end{bmatrix}$$
```

The LaTex code above, produces the output shown below,

$$\begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$$

Now one can combine the 2 different kinds of dot symbols to form a generally sized matrix,

```
$$\begin{bmatrix}
v_{11} & \dots & v_{1j} \\
\vdots & & \vdots \\
v_{i1} & \dots & v_{ij}
\end{bmatrix}$$
```

The LaTex code above, produces the output shown below,

$$\begin{bmatrix} v_{11} & \dots & v_{1j} \\ \vdots & & \vdots \\ v_{i1} & \dots & v_{ij} \end{bmatrix}$$

The center element of the matrix above is left blank, but one can populate it with \ddots to invoke diagonal dots. That would make the matrix notation look more complete,

```
$$\begin{bmatrix}
v_{11} & \dots & v_{1j} \\
\vdots & \ddots & \vdots \\
v_{i1} & \dots & v_{ij}
\end{bmatrix}$$
```

The LaTex code above, produces the output shown below,

$$\begin{bmatrix} v_{11} & \dots & v_{1j} \\ \vdots & \ddots & \vdots \\ v_{i1} & \dots & v_{ij} \end{bmatrix}$$

From all of this, the following example should be readable by now,

```
$$\begin{bmatrix}
\partial x_{1} & \partial x_{2} & \dots & \partial x_{n}
\end{bmatrix}^{T} = \begin{bmatrix}
\partial x_{1} \\ \partial x_{2} \\ \vdots \\ \partial x_{n}
\end{bmatrix}$$
```

The LaTex code above, produces the output shown below,

$$\begin{bmatrix} \partial x_1 & \partial x_2 & \dots & \partial x_n \end{bmatrix}^T = \begin{bmatrix} \partial x_1 \\ \partial x_2 \\ \vdots \\ \partial x_n \end{bmatrix}$$

0.10 Document Organization

If a document gets sufficiently large, putting all the content together without organization confuses the reader. There are a few important commands for text writing. If 2 lines of text is separated by more than a single line of space in between, LaTex would interpret that as a command to put a newline character in between the lines of text. Alternatively, one can invoke the \\ command.

If one wants more spacing between the lines, one can invoke the newline command successively like this \\~\\. The author can chain the newline command more more than just 2 times, like this: \\~\\~\\

0.10.1 Organizational Hierarchy

The LaTex typesetting system recognizes 7 levels of organizational hierarchy, those are listed below from the most general (large) to the most specific (small):

1. part: To use, invoke command \part{Enter Name of Part}
2. chapter: To use, invoke command \chapter{Enter Name of chapter}
3. section: To use, invoke command \section{Enter Name of Section}
4. subsection: To use, invoke command \subsection{Enter Name of Subsection}
5. subsubsection: To use, invoke command \subsubsection{Enter Name of subsubsection}
6. paragraph: To use, invoke command \paragraph{Enter Name of paragraph}
7. subparagraph: To use, invoke command \subparagraph{Enter Name of subparagraph}

Below shows an example of a document that contains all 7 organizational structures. You are encouraged to compile the document for yourself to see the true appearance of the document.

```
\documentclass[a4paper, 12pt]{report}

\begin{document}

\part{Engineering Problems: Fluid Mechanics}
We get to choose a subset of a wide variety of engineering problems. One of the hardest engineering
    problems without a fully definitive answer would be fluid mechanics. Fluid mechanics have been
    studied for a long time, yet scientists have failed to provide a definitive general analytical
    solution to fluid mechanics. It might be because of the chaotic nature of the fluid mechanic basic
     governing equations.

\chapter{Incompressible Flows}
Fluid Flow is often complex. Fluid flow can be thought to be divisible as compressible and
    incompressible flows. Flows which are low in relative speed can be considered incompressible
    meanwhile flows which are high in relative speed can be considered compressible. To determine
    compressibility, one has to look at the Mach number, a non-dimensional parameter that is the ratio
     of the fluid speed relative to local sound speed.

\section{Inviscid Thin Airfoil Theory}
Engineers seeking to design aircrafts and other wing-like devices must familarize themselves with the
     basics of inviscid thin airfoil theory. Inviscid thin airfoil theory relies on alot of
    assumptions and may not be very useful in the most general cases of thicker wings and higher
    angles of attack, or during stall, but inviscid thin airfoil theory can match experimental data
    very well as long as its underlying assumptions are not violated.

\subsection{Theoretical and Mathematical Basis}
Thin Airfoil theory relies on alot of mathematical operations that can be considered "expensive". To
    name a few, one must be familiar with integration. One should also be able to solve a large linear
     system of equations.

\subsubsection{Integral Operations}
The integrals in thin airfoil theory can be used to determine coefficients of A which can then be
    used to determine more useful properties like lift coefficient and moment coefficients.

\paragraph{Basic-Pre-Requisite Knowledge}
When approaching thin airfoil theory, one must be prepared to see the knowledge to be a kind of tree.
     If one is not familar with solving systems of linear equations or the integral operations that
    often plague thin airfoil theory, one must first familiarize themselves with the "basic" pre-
    requisite knowledge before proceding further. This is vital for any complicated knowledge
    disciplines.

\subparagraph{Glauret Integral}
```

```
The Glauret integral is a rather difficult integral. It also appears alot in many other areas of
    fluid mechanics, which is why different values of n in the integral expression has been carefully
    evaluated beforehand. This saves overhead when performing computations for the thin airfoil
    problem.

\end{document}
```

Personally when writing documents, I like to begin from the `section` level and expand the organizational tree to be more general or more specific on an as-needed basis. Usually, I would stop at the `subsubsection` level at the most specific and would stop at the `part` level for large documents. For Smaller documents, I would stop at the `section` level as the most general organizational structure.

0.10.2 Adding Titles

The Title commands are added after the \begin{document} command. Below shows an example of how to make titles in a new document:

```
\documentclass[a4paper, 12pt]{report}

\begin{document}

%Start of the Title Commands
\title{Comprehensive Guide to Engineering}
\author{Ginger Gengar}
\date{$15^{th}$ April 2022}
\maketitle
\newpage
%End of the Title Commands

The contents of the actual document begins here

\end{document}
```

Let us analyze the commands. The \title command specifies the main message of the title. The \author and \date commands are self explanatory. The \maketitle command tells the compiler to truly make the title. If this command is absent, the title would not be printed. This gives the author an option of "showing" or "hiding" the title of the document. To "show" the title of the document, the author just has to invoke the \maketitle command. To "hide" the title, just comment the \maketitle command out.

Try commenting out the various parts of the title command to see which parts of the title the commands correspond to.

0.10.3 Table of Contents

Table of content generation in LaTex is incredibly easy. Just invoke the command \tableofcontents wherever you wish to write the table of contents. Below shows an example of how to use the \tableofcontents command from earlier,

```
\documentclass[a4paper, 12pt]{report}

\begin{document}

%Command to Write the Table of contents
\tableofcontents

\part{Engineering Problems: Fluid Mechanics}
We get to choose a subset of a wide variety of engineering problems. One of the hardest engineering
    problems without a fully definitive answer would be fluid mechanics. Fluid mechanics have been
    studied for a long time, yet scientists have failed to provide a definitive general analytical
    solution to fluid mechanics. It might be because of the chaotic nature of the fluid mechanic basic
     governing equations.
```

```
\chapter{Incompressible Flows}
Fluid Flow is often complex. Fluid flow can be thought to be divisible as compressible and
    incompressible flows. Flows which are low in relative speed can be considered incompressible
    meanwhile flows which are high in relative speed can be considered compressible. To determine
    compressibility, one has to look at the Mach number, a non-dimensional parameter that is the ratio
     of the fluid speed relative to local sound speed.

\section{Inviscid Thin Airfoil Theory}
Engineers seeking to design aircrafts and other wing-like devices must familarize themselves with the
     basics of inviscid thin airfoil theory. Inviscid thin airfoil theory relies on alot of
    assumptions and may not be very useful in the most general cases of thicker wings and higher
    angles of attack, or during stall, but inviscid thin airfoil theory can match experimental data
    very well as long as its underlying assumptions are not violated.

\subsection{Theoretical and Mathematical Basis}
Thin Airfoil theory relies on alot of mathematical operations that can be considered "expensive". To
    name a few, one must be familiar with integration. One should also be able to solve a large linear
     system of equations.

\subsubsection{Integral Operations}
The integrals in thin airfoil theory can be used to determine coefficients of A which can then be
    used to determine more useful properties like lift coefficient and moment coefficients.

\paragraph{Basic-Pre-Requisite Knowledge}
When approaching thin airfoil theory, one must be prepared to see the knowledge to be a kind of tree.
     If one is not familar with solving systems of linear equations or the integral operations that
    often plague thin airfoil theory, one must first familiarize themselves with the "basic" pre-
    requisite knowledge before proceding further. This is vital for any complicated knowledge
    disciplines.

\subparagraph{Glauret Integral}
The Glauret integral is a rather difficult integral. It also appears alot in many other areas of
    fluid mechanics, which is why different values of n in the integral expression has been carefully
    evaluated beforehand. This saves overhead when performing computations for the thin airfoil
    problem.

\end{document}
```

0.11 Miscellaneous

0.11.1 Image Insertion

Image insertion can be done using the `graphicx` package. To load the package, use the following command,

```
\usepackage{graphicx}
```

The rest of the exmaples here assumes that an image named `image.png` is located on the same directory as the `.tex` file. The example below is a simple way to include images,

```
\\~\\\includegraphics[scale=0.3]{image.png}
```

The size of the image that is printed can be rescaled by changing the `scale` from 0.3 to say 0.4. One is free to change the scaling to make the image size display to one's preference. For most documents, this should be sufficient, but sometimes we need to show documents with captions and perhaps refer to it later. The example below shows how to do this,

```
\begin{figure}[H]\centering
\includegraphics[scale=0.3]{image.png}
\caption{This is an image}
\label{image labelling}
\end{figure}
```

The example above requires the `float` package though. That package can be loaded like any other package with the \usepackage command. Below is the command to load the `float` package:

```
\usepackage{float}
```

0.11.2 Programming Insertion

To insert programs, one can use the `listings` package. To load the `listings` package,

```
\usepackage{listings}
```

To include direct computer code, we can start a `listings` environment. Here is an example of including a program inside a `LaTex` document,

```
\documentclass[a4paper, 12pt]{report}

\usepackage{listings}

\begin{document}

Below is a listing of a simple C program:
\begin{lstlisting}[language=C]
#include <stdio.h>
int main(void){
printf("Hello world\n");
return 0;}
\end{lstlisting}

\end{document}
```

The resulting listing however looks quite ugly. So there are a few parameters we can change inside the `listings` package that can make the listings look nicer. Here is an improved version:

```
\documentclass[a4paper, 12pt]{report}

%Package Loading
\usepackage{listings}
\usepackage{xcolor}

%Definition of various colors
\definecolor{codegreen}{rgb}{0,0.6,0}
\definecolor{codegray}{rgb}{0.5,0.5,0.5}
\definecolor{codepurple}{rgb}{0.58,0,0.82}
\definecolor{backcolour}{rgb}{0.95,0.95,0.92}

%Changing Parameters within listings package
\lstset{
  columns=fullflexible,
  frame=single,
  breaklines=true,
  backgroundcolor=\color{backcolour},
  commentstyle=\color{codegreen},
  keywordstyle=\color{magenta},
  numberstyle=\tiny\color{codegray},
  stringstyle=\color{codepurple},
  basicstyle=\ttfamily\footnotesize,
  keepspaces=true,
  numbersep=5pt,
  showspaces=true,
  showtabs=true,
  tabsize=2
}

\begin{document}
```

```
Below is a listing of a simple C program:
\begin{lstlisting}[language=C]
#include <stdio.h>
int main(void){
printf("Hello world\n");
return 0;}
\end{lstlisting}

\end{document}
```

It is important to note that the example above uses a new package, `xcolor`. `xcolor` is used to describe colors such as `codegreen` and `codegray` and so on.

0.12 Additional References

`Overleaf` is a fantastic rescource containing very beginner-friendly explanations on how to write in `LaTex`. `Overleaf` also contains a web-based application that allows `LaTex` document to be written through the web-browser. The link for `Overleaf`:

`https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes`

The official `LaTex` documentation can be found in the link below:

`https://www.latex-project.org/help/documentation/`

There are many `LaTex` packages out there. The documentation for many of these packages can be found in `CTAN` (Comprehensive Tex Archive Network). The link to that is shown below:

`https://ctan.org/?lang=en`

These are my personal preference for packages and their uses:

1. `amsmath`: Various mathematical utilities
2. `comment`: Allows the usage of comment blocks
3. `amssymb`: For more mathematical symbols
4. `esint`: For notations of double and triple integrals
5. `geometry`: To trim the margin of the document and also declare the layout, like landscape or portrait
6. `graphicx`: To include images in a `LaTex` document
7. `hyperref`: Generate hyperlinks on the `LaTex` document to online sites.
8. `listings`: A way to include computer programming languages in a `LaTex` document seamlessly
9. `xcolor`: A way to incorporate colors to `LaTex`. Usually used for syntax highlighting with `listings`
10. `array`: Extends the usage of tables. Table column sizes can be set through this package
11. `multicol`: Allows the document to be written in columns
12. `circuitikz`: Allows the drawing of circuit diagrams
13. `tikz`: A general drawing tool for `LaTex`