Investigation of System Architecture, Databases, and Caching systems For backend web development

Independent Study Thesis

Presented in Partial Fulfillment of the Requirements for
the Degree Bachelor of Arts in the
MCS at The College of Wooster

by
Jungho Park

The College of Wooster
2026

Advised by:

Drew Guarnera (Computer Science)

THE COLLEGE OF WOOSTER

Abstract

Include a short summary of your thesis, including any pertinent results. This section is *not* optional, and the reader should be able to learn the meat of your thesis by reading this (short) section.

This work is dedicated to the future generations of Wooster students.

Acknowledgments

I would like to acknowledge Prof. Lowell Boone in the Physics Department for his suggestions and code.

Vita

Publications

Fields of Study Major field: Major

Minor field: Minor

Specialization: Area of IS research

Contents

List of Figures

List of Tables

List of Listings

Preface

The purpose of this document is to provide you with a template for typesetting your IS using LaTeX. LaTeX is very similar to HTML in the sense that it is a markup language. What does this mean? Well, basically it means you need only enter the commands for structuring your IS, i.e., identify chapters, sections, subsections, equations, quotes, etc. You do not need to worry about any of the formatting. The `woosterthesis` class takes care of all the formatting.

Here is how I plan on introducing you to LaTeX. The Introduction gives some reasons for why one might find LaTeX superior to MS Word™. Chapter **??** will demonstrate how one starts typesetting a document and works with text in LaTeX. Chapter **??** discusses the creation of tables and how one puts figures into a thesis. Chapter **??** talks about creating a bibliography/references section and an index. There are three Appendices which discuss typesetting mathematics and computer program code. The Afterword will discuss some of the particulars of how a LaTeX document gets processed and what packages the `woosterthesis` class uses and are assumed to be available on your system.

Hopefully, this document will be enough to get you started. If you have questions, please refer to **mgbcr04**, **kd03**, **ophs03**, **feu02**, **fly03**, or **gra96**.

CHAPTER 1

Introduction

So why would you want to use LaTeX instead of Microsoft Word™? I can think of several reasons. The main one for this author is that LaTeX takes care of all the numbering automatically. This means that if you decide to rearrange material in your IS, you do not have to worry about renumbering or references. This makes it very easy to play around with the structure of your thesis. The second reason is that it is ultimately faster than Word™. How? Well, after a week or so of using LaTeX , you will begin to remember the commands that you use frequently and won't have to use the LaTeX pallet in TeXShop or TeXworks. So, you can just type everything including the mathematics, where with Word™ you would have to use the Equation Editor.

I have also tried to make things more efficient by organizing the example folder as follows. There is a `main.tex` file which is what you will enter all the information about your IS into and is the document you will typeset. `main.tex` also has explanations about other files that you might need to edit. In addition, there are folders for chapters, appendices, styles, and figures. This structure is there to try and reduce file clutter and to help you stay organized. There should also be a .bib file which you can use as a model for your own .bib file. The .bib file has your bibliographic information.

LaTeX is easy to learn. For an average IS, the author will only need to learn a handful of commands. For this small bit of effort, you get a tremendous amount of flexibility and a very beautiful document. The following chapters will introduce some of the common things a student might need to do in a thesis.

1.1 What is in `main.tex`?

Before we move on let's talk a little bit about what is at the beginning of `main.tex`. The file starts with `\documentclass{woosterthes` which must be at the beginning of every IS. In the brackets are options for the woosterthesis class. The options are the same as for the `book` class with some additional options `abstractonly`, `acs`, `alltt`, `apa`, `blacklinks`, `chicago`, `citeorder`, `code`, `colophon`, `dropcaps`, `euler`, `foreignlanguage`, `guass`, `index`, `kaukecopyright`, `maple`, `minimalstyle`, `mla`, `nostyle`, `palatino`, `picins`, `scottie`, `singlespace`, `tikz`, `verbatim`, `wblack`, and woostercopyright. If no options are specified then the class default options letterpaper, 12pt, oneside , onecolumn , final, and openany are used.

The abstractonly option will allow you to print just the Abstract. The acs option implements the American Chemical Society citation and reference style. The alltt option loads the alltt package for using typewriter type in various ways and the apa option implements the APA citation and reference style. The blacklinks option will make the hyperlinks in the PDF version of the thesis black and suitable for printing; normally the links are colored to provide visual clues to the reader. The chicago option implements Chicago style citation and references. The citeorder option orders the references according to the citation order of the IS. The code option will use listings style to format program code examples. The colophon option will include a colophon which is a section that describes the fonts and other settings used to produce the manuscript. dropcaps loads the lettrine package for doing dropped capitals and the euler and guass options load the woofncychap package with the named option which will change the look of chapter headings. The foreignlanguage option will load the csquotes package and either the polyglossia or babel package depending on if XƎTEX is being used to allow the input and formatting of sections of text in a foreign language. The index option will allow the makeidx package to be loaded so that if you have index entries they will be added to an index (this reqires additional steps). The kaukecopyright option will put the Kauke Hall symbol with the pre 2021 wordmark on the copyright page. The maple option will load the Maple package for including Maple code. The minimalstyle option will use custom styling for the Table of Contents and List of Figures, Tables, and Listings and otherwise revert to the default book class styles. The mla option implements the MLA citation and reference style. The nostyle option will remove all custom styling and revert to the default book class styles. The palatino option will use the pxfonts package which uses the Palatino fonts. The picins option will use the wrapfig package to allow text to wrap around images and verbatim allows one to set verbatim what is entered. The tikz option loads the Ti*k*Z package enabling users to draw figures in their LATEX document. The singlespace option will use the setspace package to typeset the document in singlespace. The wblack option includes an opaque Wooster ”W” (as of 8/2021) in the background of the title page instead of the default opaque Kauke Hall image, the scottie option includes an opaque grayscale image of the Scottie mascot in the background of the title page instead of the default opaque Kauke Hall image, and the woostercopyright option includes a copyright notice with the new (as of 8/2021) Wooster wordmark (see Appendix **??** for images of what these options produce). Adding or deleting options from the comma separated list will change the appearance of the document and some options should only be used after consulting your advisor. Now let’s move on to some other things that you’ll need to deal with: text, figures, pictures, and tables.

CHAPTER 2

Software Architecture Style

There are multiple architectural styles that could be applied when designing the backend system of a web application. The two main categories are monolithic and distributed. Subcategories such as layered services, microservices, event-driven services, are all under these two architecture styles. Each of these design have the pros and cons.

2.1 Monolithic Applications

A monolithic application is where all the logical components of the application is deployed as one unit. This also means that the application will be ran in one process. For example, if there is a stand-alone Python Flask application deployed to a server, this is monolithic applications. The general pros and cons of this architecture style are the following:

Pros: - Simplicity: Typically, monolithic applications have a single codebase, which makes them easier to develop and understand - Cost: Monolithis are cheaper to build and operate because they tend to be simpler and require less infrastructure. - Feasibility: Monoliths are simple and relatively cheap, freeing developers to experiment and deliver systems faster. - Reliability: Monoliths makes few or no network calls, which usually means more reliable application. - Debuggability: If a bug is spotted or get an error stack trace, debugging is easy, since all the code is in one place. Cons: - Scalability: - Evolvability: Making changes to monolithic applications become harder as it grows. Since the whole application is one codebase, it is not possible to adapt different technology stacks to different domains if needed. - Reliability: Since monolithic applications are deployed as a single unit, any bug that degrades the service will affect the whole application. - Deployability: Implementing any change will require redeploy8ing the whole application, which could introduce a lot of risk.

2.1.1 Layered Architecture

Layered architecture is a sub category of the Monolithic architecture. Applications that are designed with this architecture has three layered parts: the presentation layer, the workflow layer, the persistence layer, and the database layer.

Each layer has it's own tasks and separated within one application. Presentation Layer: The presentation layer is where the UI is displayed and where the users interact with the system. All components that are related to the UI will be included in this layer. Workflow Layer: The workflow layer consists all code that are related to logic such as business logic, workflows, and validations. This is where most of the application's code is contained. Persistence Layer: The persistence layer encapsulates the behavior that is needed to make objects persistent, such as mapping the architecture to code-base hierarchies into set-based relational databases. Database Layer: The database layer is optional. This layer includes the database, or some kind of way to persist information.

Each of the layer may contain multiple problem domains. For example, for a restaurant, there could be domains like place order, deliver order, manage recipes, or mange inventory. Different problem domains could exist together in each layer of the application.

An implication of the layered monolithic architecture is the MVC design pattern. In MVC, the model represents business logic and entities in the application; the view represents the user interface; and the controller handles the workflow, stitching model elements together to provide the application's functionality, as shown here: (head first software architecture chapter 6)

General Advantages of Layered Architecture: - Feasability (Simplicity) - Technical Partitioning - Data-intensive - Performance (if well designed) - Quick to build - General Disadvantages of Layered Architecture - Deployability - Complexity - Scalability: Non flexibility in change in problem dodmain - Elasticity - Testability Some applications using the layered architecture may have different structure from the original. Some layers might be separated, such as the presentation layer being separated from the other layers. Each structure will have advantages and disadvantages within the layered architecture.

2.1.2 Modular Monolith

Modular monolith is when an application is divided into different modules within the monolithic system. A module is one domain of the application, and each domain will have its independent code base. For instance, an application for a restaurant might have different domains such as orders, deliveries, or recipes. In this case, there could be separate databases for each modules, or multiple schemas within a single database. It is important to keep in mind that only the code base is independent. Pros: - Domain partitioning: each domain will be implemented in a different module, which allows to build teams that specializes in one or more of these domains. - Performance: Since there are no network calls within the application like other monolithic applications, performance is good. - Maintainability: Each module are separate from each other. - Testability: Cons: - Hard to reuse: Since each module have its own code, it is hard to reuse the logic and utilities across the modules - Single set of architecture: Since all the code is within one application, logics and databases could become complicated, making it hard to scale.

2.1.3 Microkernal Architecture

The microkernel architecture is consisted with a core of the application, and plugins. The core of the application is where the main application is being run, and the plugins are connected to the core application. It is used when a lot of customization is needed for the software

2.2 Distributed Systems

Distributed architecture is when the logical componnents of the application are split up into multiple units. These units each run in their individual process and communicate with each other over the network. This architecture style encourage loose coupling of each services. The general pros and cons of this architecture style are the following:

Pros: - Scalability: Distributed architectures deploy different logical components separately from one another, so it is easy to add new services. - Modularity: Distributed architecture encourage a high degree of modularity because their logical components must be loosely coupled. - Testablility: Each deployment only serves a select group of logical components. This makes testing a log easier-even as the application grows. - Deployability: Distributed architectures encourage lots of small units. They evolved after modern engineering principles like continuous integration, continuous deployments, and automated testing became the norm. - Fault Tolerance: Even if one piece of the system fails, the rest of the system can continue functioning. Cons: - Performance: Distributed architectures involve lots of small services that communicate with each other over the network. This can affect performance, but there are ways to improve this. - Cost: More servers are needed to deploy multiple units. These services will need to talk to each other, which entails setting up and maintaining network infrastructure. - Simplicity: Distributed systems are complicated to understand from how they work to debugging errors. - Debuggability: Errors could happen in any unit involved in servicing a request. Since logical components are deployed in separate units, tracing errors could become complicated.

There are both pros and cons for monolithic and distributed systems and choosing the appropriate architecture for each situation is crucial. After deciding which of the two main architecture will be used, the subcategories shows the more detailed ways of designing and structuring each units in the application(s).

2.2.1 Microservices Architecture

A microservice is a service that is separately deployed unit of software that performs some business or infrastructure process.(chapter 10). A microservices architecture is part of the layered system, where microservices communicate which each other to make an application. Since the system is divided into multiple parts, it is essential how to divide the application, such as deciding how small or how big each microservices could get into. Some of the factors that is considered to make microservices smaller are the following: cohesiveness, fault tolerance, access control, code volatility, and scalability. If a part of the software has lack of cohesiveness and loosely coupled, it is a good idea to separate it into smaller microservices. This will allow higher scalability. If a certain part of the application produces fatal errors, having those part in a sparate microservice will decrease the probability to shut down the whole system. For security and authentication, it is important to have these access controlabilities into a single microservice, so it does not get too complicated when managing the information. Finally, if one part of the microservice change, or scale faster than the others, it is good to consider to have a separate microservice for that application, since testing the entire microservice would be much more challenging compared to a small portion of the microservice. On the other hand, there are times when it is encouraged to make the microservices bigger: Database transactions, data dependencies, and workflow. It is not possible to perform a single database commit or rollback when a request involves multiple microservices. For data consistency and integrity, it is important to combine functionalities that require these kind of behavior into a single microservice. If a part of a microservice has highly coupled data, such as when a database table refers to the key of another database table, it is better to keep these functionalities as a single microservice, to keep the data integrity of the database. If a single request requires separate microservices to communicate with each other, this request is coupled. If too much coupling is occurred between microsesrvices, there are many negative effects. For example, performance is affected by network, security, and data latency. Scalability is affected because each microservice in the call chain must scale as the other microservices scale (something that is hard to coordinate). Fault tolerance is affected because if one of the microservices in the chain becomes unresponsive or unavailable, the request cannot be processed. It is good practice to consider the workflow and decide whether to keep the microservice big.

Balance

Sharing functionalities

There are many times when the same code has to be used in multiple microservices. There are mainly two ways for sharing code when building a microservice architecture. Creating a shared service, or a shared library. A shared service is a separate microservice that contains a shared functionality that other microservices can call remotely. The advantages of using this is that eventhough a code is changed in the shared microservices, code in other microservices are not required to change. Also, this shared service could be written in any language, which is useful when microservices are implemented in multiple languages. A disadvantage of using a shared service is coupling that happens between the microservices and the shared service. This leads to risks when changing a shared service since it can affect the other microservices that call it. Furthermore, when the shared service is down for some reason, the microservices that require the shared service would not function. Another disadvantage is network latency, …. A shared library is a more common way for code reuse. A library will be built including all the code that are reused in different microsesrvices, and once they microservices are deployed, each microservice will have all the shared functionality available. The biggest advantage of using a shared library is that network performance and scalability is better than shared service, since it is not remote, but code is included in the compile time of the microservice. However, multiple shared libraries will be needed if microservices are written in different programming languages. Also, managing dependencies between microservices and shared libraries could be a challenge if there are multiple microservices using the shared code.

Workflow

Microservices communicate with each other to make the whole application. It is important to know how these are connected, and this is where workflow comes in. The term workflow means when two or more microservices are called for a single request. There are two ways to handle workflows: centralized workflow management, or decentralized workflow management.

Centralized workflow management:Workflow

This workflow management style is where there is a microservice that coordinates all the microservices that are needed to handle a certain single request. This microservice will be responsible for calling all related microservices, knowing the current state of the workflow and what happens next, summarize all data from each microservice, and handling errors. The advantage of this workflow management style is that the order of microservices for each requests are clear. The central microservice always has the exact routes the request hast to take, which allows to track the status where each requests stopped and where to restart. This allows to handle errors efficiently. Also, it is easy to change a workflow since this all changes in one central microservice The disadvantage is that tight coupling between the central microservice and the other microservices. This can lead to lower scalability, since changing a microservice will affect the central microservice. Moreover, performance might get delayed since the central microservice is indeed another microservice that requires remote calls and it saves the workflow state data in a database which slows down the performance.

Decentralized workflow management

The decentralized workflow does not have a central microservice, but rather all microservices redirect to another microservice by the given request until the request is complete. It is important to not use one of the microservices as a central microservice. The advantage of using this workflow pattern is that it is loosely coupled compared to the central workflow management, which has better scalability. Also, since the microservices does not have to connect back to the central microservice, it has better responsiveness, meaning less delay and better performance. On the other hand, this pattern lacks in error handling, since each microservice is responsible for managing the error workflow, which could lead to too much communication between services. Also, because there is not a central microservice that has the order of microservices that needs to be called, it is hard to tell which state the response is at, which decreases the ability to recover when the request is delayed and needs to be restarted again.

Pros and Cons of microservice architecture. There are moments where the microservice architecture are needed when choosing the software design. There are multiple advantages when choosing this structure. Loose coupling is a huge advantage. Since each microservices are single-purpose and they are deployed separately, and easy to maintain.

This means that it is easy to change a particular function that needs modification. Also, it is easy to test the application, as the scope is much smaller compared to the monolithic architecture. Fault tolerance is another advantage that comes from loose coupling. Even if a particular microservice fails, it does not break the whole system. Furthermore, microservice architecture is easy to scale and evolve since we just have to add or modify a microservice related to the area.

However, there are also disadvantages of using the microservice architecture. This architecture is complex. It requires multiple decisions(the aspects discussed above: workflow, shared code etc..) depending on the situation. Since it is complex, as microservices communicate with each other, the performance also decreases. The request might have to wait for the network or undergo additional security checks and make extra database calls. Lastly. deploying all of these microservices will increase the cost of the entire application.

2.2.2 Event-Driven Architecture(EDA)

What is an Event An event in computer science is a way for a service to let the rest of the system know that something important has just happened. In EDA, events are the means of passing information to other services (head first chapter11). The event contains data and it is broadcasted to services using topics that are connected to the sender of the event. However, events are asynchronous, which means that although the services are connected, the sender of the event does not wait on the response of the receiving service. This is the key difference between a event and message. Messages includes a command or some kind of a request for the receiving service, and the service sending out the messages require a response, making it synchronous. Messages are also sent to only a single service using queues.

Asynchronous vs Synchronous Asynchronous communications do not need to wait for the response, even though the receiving services are available or not. These kinds of communications are also called Fire and Forget. On the other hand, synchronous communication needs to stop and wait until the response, which means that the service in response must be available. EDA relies on the asynchronous communication when sending and receiving events.

Advantages, Disadvantages of Asynchronous Communications Asynchronous communications has advantages in responsiveness compared to synchronous communications. Since responses by the receiving services are unnecessary, it takes less time to complete a request. However, this is also a crucial disadvantage. Since we do not know if the receiving services have successfully completed the request, it is prone to error handling. Here is a diagram.

Advantages of EDA Event Driven Architecture is highly decoupled, which makes all services independent and easy to maintain. Furthermore, the asynchronous communication increases the performance of the application. Since EDAs are highly decoupled with this type of communication, it is easy to scale and evolve. Disadvantages of EDA Similar to microservices, EDAs are complex. Deciding which database topology for the architecture, asynchronous communications and parallel event processing makes adds complexity to the application. Also, asynchronous communication makes it more difficult to test out the program. Since the request is proceeded without any response or synchronous calls, the context of the test is vague. If the application needs multiple synchronous calls, EDA is not the right choice for the product.

CHAPTER 3

Working with figures and tables

3.1 Getting a simple figure in the document

In this chapter we want to talk about including figures and tables in the document. To insert a simple figure, you can enter something like

```
\begin{figure}[!ht]
\begin{center}
\woopic{picture3}{.8}
\end{center}
\caption{Our first
 picture}\label{first}
\end{figure}
```

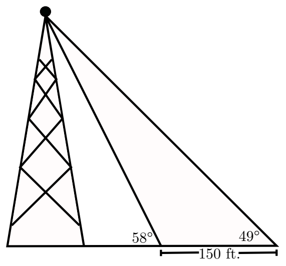


Figure 3.1: Our first picture

The `!ht` tell LaTeX to try and place the figure here no matter what or at the top of the next page. The `\woopic` command takes the name of the picture as the first argument and the scaling factor as the second argument. The scaling factor must be between zero and one and the figure name must have *no spaces*. Your figures can be in one of four formats: jpg, png, tif, or pdf. Captions are placed below the figure and your label should be placed after the caption.

In the next example we are using the woosterthesis option `picins` to typeset a picture inside a paragraph and have the text wrap around the figure. This option loads the `wrapfig` package. One thing to note is that the figures placed in this manner do not float with the other figures and as such numbering could get out of sequence. Keep an eye out for such behavior. This technique should be used sparingly in your thesis.

```
\newcommand{\sample}{Some text that is reused over and over
 again in the example. }
```

```
\begin{wrapfigure}{r}{2.2in}
\woopic{picture2}{.4}
\caption{Conchoid.}
\end{wrapfigure}
\sample\sample\sample\sample
```

Some text that is reused over and over again in the example. Some text that is reused over and over again in the example. Some text that is reused over and over again in the example. Some text that is reused over and over again in the example. Some text that is reused over and over again in the example. Some text that is reused over and over again in the example. Some text that is reused over and over again in the example. Some text that is reused over and over again in the example. Some text that is reused over and over again in the example.

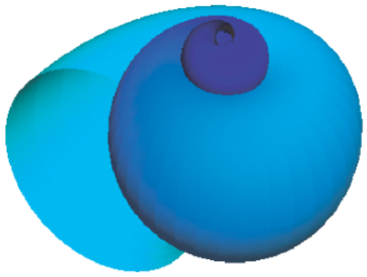


Figure 3.2: Conchoid.

3.1.1 Drawing figures in LaTeX

It is also possible to use the `TikZ` package to draw figures like Figures **??** or **??** in your document. Ti*k*Z/PGF are a pair of languages that allow users to create vector graphics as part of their LaTeX document using commands that are TeX macros. It is possible to create extremely complex scalable graphics using Ti*k*Z, see the Ti*k*Z wikipedia entry for some examples.

```
\begin{figure}[!ht]\centering
\begin{tikzpicture}[
roundnode/.style={circle, draw=green!60, fill=green!5, very thick, minimum size=7mm},
squarednode/.style={rectangle, draw=red!60, fill=red!5, very thick, minimum size=5mm},
]
%Nodes
\node[squarednode]      (maintopic)                              {2};
\node[roundnode]        (uppercircle)       [above=of maintopic] {1};
\node[squarednode]      (rightsquare)       [right=of maintopic] {3};
\node[roundnode]        (lowercircle)       [below=of maintopic] {4};
%Lines
\draw[->] (uppercircle.south) -- (maintopic.north);
\draw[->] (maintopic.east) -- (rightsquare.west);
\draw[->] (rightsquare.south) .. controls +(down:7mm) and +(right:7mm) .. (lowercircle.east);
\end{tikzpicture}
\caption{Tree diagram}\label{tree}
\end{figure}
```

```
\begin{figure}[!ht]\centering
\begin{tikzpicture}
\draw[gray, thick] (-1,2) -- (2,-4);
\draw[gray, thick] (-1,-1) -- (2,2);
```

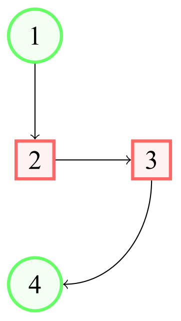


Figure 3.3: Tree diagram

```
\filldraw[black] (0,0) circle (2pt) node[anchor=west]{Intersection point};
\end{tikzpicture}
\caption{Intersecting lines}\label{lines}
\end{figure}
```

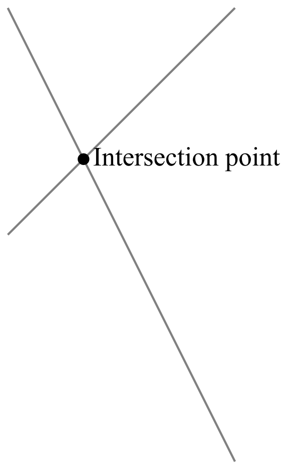


Figure 3.4: Intersecting lines

3.1.2 Minipages

You can also create minipages in your documents to accomplish more complicated formatting. For example, you could try the following which produces Figure **??**.

```
\begin{minipage}[t][3 in][t]{1 in}
This is a minipage which is 3 in tall and 1 in wide.
 Top Text Text Text Text.\end{minipage}\hfill
\begin{minipage}[t][3 in][c]{1 in}
This is a minipage which is 3 in tall and 1 in wide.
 Center Text Text Text Text.\end{minipage}\hfill
\begin{minipage}[t][3 in][b]{1 in}
This is a minipage which is 3 in tall and 1 in wide.
 Bottom Text Text Text Text.\end{minipage}
```

This is a minipage which is 3 in tall and 1 in wide. Top Text Text Text Text.

This is a minipage which is 3 in tall and 1 in wide. Center Text Text Text Text.

This is a minipage which is 3 in tall and 1 in wide. Bottom Text Text Text Text.

Figure 3.5: Minipage example

In the example above, the syntax `\begin{minipage}[t][3 in][t]{1 in}` follows the convention

`\begin{minipage}[minipageposition][height][textposition]{width}`

3.1.3 How to get more than one picture in the same figure

You can use minipages to put more than one picture in a figure. Here is an example of how to do this.

```
\begin{minipage}[!ht]{6cm}
\woopic{picture1}{.4}
\par
\caption[What goes in the List of Figures]{Left}
\end{minipage}
\hfill
\begin{minipage}[!ht]{6cm}
\woopic{picture2}{.4}
\end{picture}\par
\caption{Right}
\end{minipage}
```

You can also use the `subfig` package to do this.

```
\begin{figure}[!ht]\centering
\subfloat[What goes in the List][Large conchoid]
{\woopic{picture1}{.4}\label{fig3:left}}
\qquad
\subfloat[What goes in the List][Small conchoid]
{\woopic{picture2}{.4}\label{fig3:right}}
\caption{Two pictures in one figure}\label{fig3}
\end{figure}
```

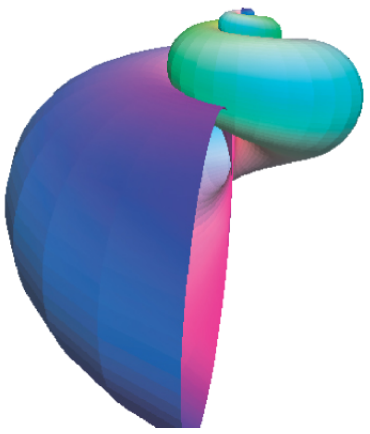


Figure 3.6: Left

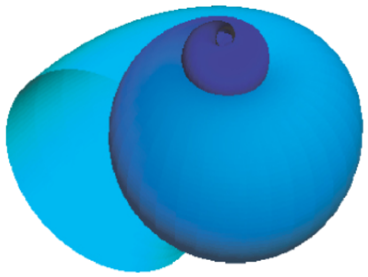
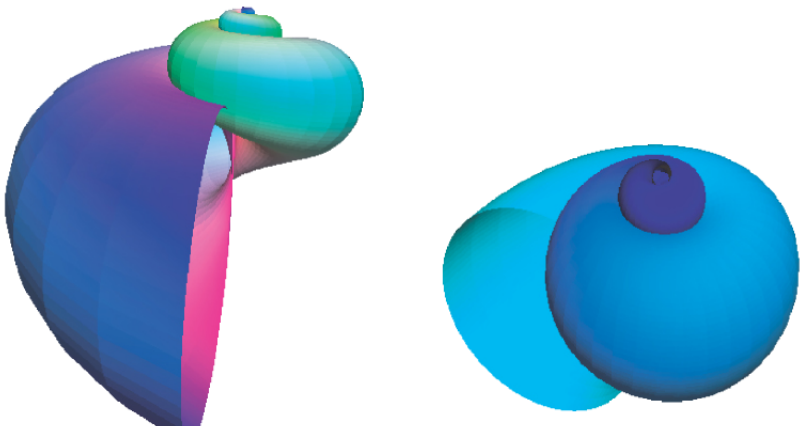


Figure 3.7: Right



(a) Large conchoid **(b)** Small conchoid

Figure 3.8: Two pictures in one figure

We should now be able to refer to either Figure **?? ??** or Figure **?? ??** using the labels we gave to the left and right images.

The reader is referred to Chapters 8, 9, and 16 of **kd03** or to Chapters 6 and 10 of **mgbcr04** for a complete discussion of figures and graphics.

3.2 Tables

Tables are easy to set up. Here is a simple table:

```
\begin{table}[!ht]
\begin{center}
\caption{Our first table}
```

```
\begin{tabular}{r l}
  $\underline{\textnormal{District}}$ &
  $\underline{\textnormal{Population}}$\\
   Applewood & 8280 \\
   Boxwood & 4600  \\
   Central & 5220
   \end{tabular}\caption{Our first table}
   \end{center}
\end{table}
```

Table 3.1: Our first table

District	Population
Applewood	8280
Boxwood	4600
Central	5220

In `\begin{tabular}{r l}` the two "r" and "l" indicate that we have two columns with right and left aligned entries and no lines dividing cells or around the table. I can make the table look more like a spreadsheet by doing:

```
\begin{table}[!ht]
\begin{center}
\caption{Our first table again}
\begin{tabular}{|r|l|}
\hline
  {\textnormal{District}} &
  {\textnormal{Population}}\\ \hline
   Applewood & 8280 \\ \hline
   Boxwood & 4600  \\ \hline
   Central & 5220\\ \hline
   \end{tabular}
   \end{center}
\end{table}
```

Table 3.2: Our first table again

District	Population
Applewood	8280
Boxwood	4600
Central	5220

Here is a more complicated example of a table.

```
\begin{table}[!ht]
\caption{Reduction of curvature by each reprojection method\label{tbl:kreduce}}
\centerline{
\begin{tabular}{|l||r|r|r|r|} \hline
\emph{Reprojection} & \multicolumn{3}{|c|}{\emph{Largest
 Reduction of Curvature}}
  & \emph{Average} \\ \cline{2-4}
\emph{Method} & \emph{Original} & \emph{Reprojected} &
 \emph{at} &
  \emph{Reduction} \\
 & \emph{Curvature} & \emph{Curvature} &
  \emph{Rotation} & \emph{of Curvature} \\
  \hline \hline
ZEEL & 0.0358 & 0.0245 &
 $\degree{45}$ & 0.0050 \\ \hline
ZEEL ext.\ & 0.0358 & 0.0245 &
 $\degree{45}$ & 0.0059 \\ \hline
Regridding & 0.0428 & 0.0166 &
 $\degree{75}$ & 0.0159 \\ \hline
Block & 0.0358 & 0.0103 &
 $\degree{45}$ & 0.0163 \\ \hline
\end{tabular}}
\end{table}
```

Table 3.3: Reduction of curvature by each reprojection method

Reprojection Method	*Largest Reduction of Curvature*			*Average*
	Original Curvature	*Reprojected Curvature*	*at Rotation*	*Reduction of Curvature*
ZEEL	0.0358	0.0245	45°	0.0050
ZEEL ext.	0.0358	0.0245	45°	0.0059
Regridding	0.0428	0.0166	75°	0.0159
Block	0.0358	0.0103	45°	0.0163

Please refer to Chapter 6 of **kd03** for a complete discussion of tables and tabular environments.

CHAPTER 4

Working with bibliographies and indicies

I would highly recommend that you use BibLaTeX for your bibliography, and that is what this class uses as of August 2022. BibLaTeX supports foreign languages and UTF-8 and provides several styles for formatting references (ACS, Chicago, APA, Turabian). For many people it also is easier to customize than BibTeX with natbib, should you need to customize the formatting of references. BibLaTeX also can make use of the Biber reference processing application which provides support for foreign languages and more sophisticated sorting options than BibTeX. You still process a special .bib file. The .bib file is where you enter your bibliographic information. Sample entries look something like

```
@article{feu02,
author= {Thomas Feuerstack},
title= {Introduction to pdf{\TeX{}}},
journal= {TUGboat},
volume= {23},
pages= {329--334},
number= {3/4},
url= {http://www.tug.org/TUGboat/Articles/tb23-3-4/tb75feu.pdf},
year= 2002}
```

or

```
@book{mgbcr04,
author= {Frank Mittelbach and Michel Goossens and
Johannes Braams and David Carlisle and Chris Rowley},
title= {The \LaTeX\ Companion},
publisher= {Addison Wesley Professional},
edition= {2nd},
address= {New York},
year= 2004}
```

For a Web site I would recommend the following

```
@misc{brei04,
author =  {Jon Breitenbucher},
title =  {{W}ooster related {L}a{T}e{X} files},
url =  {https://woolatex.spaces.wooster.edu},
howpublished= {World Wide Web},
year= 2021,
note =  {Accessed on 09/27/2021}}
```

You can make a reference by typing `\citet{mgbcr04}` to produce **mgbcr04**. Other forms for citation include `\citep{mgbcr04}` or `\citeauthor {mgbcr04}` to produce [**mgbcr04**] or **mgbcr04** respectively. You can consult **kd03** or **mgbcr04** to find out how to format entries in the .bib file and what options each reference type has.[1]

Indicies are also relatively easy to create. If I wanted to have Wooster show up in the index, I would enter `Wooster\index{Wooster}` in my source file. I could create a subentry for User Services by entering `User Services\index{Wooste` A subsubentry for Help Desk would be entered as `\index{Wooster!User Services!Help Desk}`.

To create the index, one needs to make sure to uncomment the `\makeindex` command in the `main.tex` file. One also needs to uncomment the makeidx entry in the `styles/packages.tex` file and then run the Makeindex program. Consult **kd03** or **mgbcr04** for further information.

[1] You could also use footnotes if your department called for that.

APPENDIX A

Typesetting Mathematical Formulae

This appendix is taken from **ophs03** under the GNU open-source documentation license. This appendix addresses the main strength of TEX: mathematical typesetting. But be warned, this appendix only scratches the surface. While the things explained here are sufficient for many people, don't despair if you can't find a solution to your mathematical typesetting needs here. It is highly likely that your problem is addressed in AMS-LATEX[1] or some other package.

A.1 General

LATEX has a special mode for typesetting mathematics. Mathematical text within a paragraph is entered between `\(` and `\)`, between `$` and `$` or between `\begin{`math`}` and `\end{math}`.

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach:
$c^{2}=a^{2}+b^{2}$
```

Add $a$ squared and $b$ squared to get $c$ squared. Or, using a more mathematical approach: $c^{2}=a^{2}+b^{2}$

```
\TeX{} is pronounced as
$\tau\epsilon$.\\[6pt]
100~m$^{3}$ of water\\[6pt]
This comes from my $\heartsuit$
```

TEX is pronounced as $\tau\epsilon$.

100 m$^{3}$ of water

This comes from my $\heartsuit$

It is preferable to *display* larger mathematical equations or formulae, rather than to typeset them on separate lines. This means you enclose them in `\[` and `\]` or between `\begin{`displaymath`}` and `\end{displaymath}`. This produces formulae which are not numbered. If you want LATEX to number them, you can use the equation environment.

[1]`CTAN:/tex-archive/macros/latex/packages/amslatex`

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach:
\begin{displaymath}
c^{2}=a^{2}+b^{2}
\end{displaymath}
And just one more line.
```

Add a squared and b squared to get c squared. Or, using a more mathematical approach:

$$c^2 = a^2 + b^2$$

And just one more line.

You can reference an equation with `\label` and `\ref`

```
\begin{equation} \label{eq:eps}
\epsilon > 0
\end{equation}
From (\ref{eq:eps}), we gather
\ldots
```

$$\epsilon > 0 \tag{A.1}$$

From (**??**), we gather …

Note that expressions will be typeset in a different style if displayed:

```
$\lim_{n \to \infty}
\sum_{k=1}^n \frac{1}{k^2}
= \frac{\pi^2}{6}$
```

$\lim_{n \to \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}$

```
\begin{displaymath}
\lim_{n \to \infty}
\sum_{k=1}^n \frac{1}{k^2}
= \frac{\pi^2}{6}
\end{displaymath}
```

$$\lim_{n \to \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}$$

There are differences between *math mode* and *text mode*. For example, in *math mode*:

1. Most spaces and linebreaks do not have any significance, as all spaces either are derived logically from the mathematical expressions or have to be specified using special commands such as `\,` , `\quad`, or `\qquad`.

2. Empty lines are not allowed. Only one paragraph per formula.

3. Each letter is considered to be the name of a variable and will be typeset as such. If you want to typeset normal text within a formula (normal upright font and normal spacing) then you have to enter the text using the `\textrm{...}` commands.

```
\begin{equation}
\forall x \in \mathbf{R}:
\qquad x^{2} \geq 0
\end{equation}
```

$$\forall x \in \mathbf{R}: \qquad x^{2} \geq 0 \tag{A.2}$$

```
\begin{equation}
x^{2} \geq 0\qquad
\textrm{for all }x\in\mathbf{R}
\end{equation}
```

$$x^2 \geq 0 \qquad \text{for all } x \in \mathbf{R} \tag{A.3}$$

Mathematicians can be very fussy about which symbols are used: it would be conventional here to use 'blackboard bold', bold symbols which is obtained using `\mathbb` from the package `amsfonts` or `amssymb`.

The last example becomes

```
\begin{displaymath}
x^{2} \geq 0\qquad
\textrm{for all }x\in\mathbb{R}
\end{displaymath}
```

$$x^2 \geq 0 \qquad \text{for all } x \in \mathbb{R}$$

A.2 Grouping in Math Mode

Most math mode commands act only on the next character. So if you want a command to affect several characters, you have to group them together using curly braces: `{...}`.

```
\begin{equation}
a^x+y \neq a^{x+y}
\end{equation}
```

$$a^x + y \neq a^{x+y} \tag{A.4}$$

A.3 Building Blocks of a Mathematical Formula

In this section, the most important commands used in mathematical typesetting will be described. Take a look at **kd03** for a detailed list of commands for typesetting mathematical symbols.

Lowercase Greek letters are entered as `\alpha`, `\beta`, `\gamma`, ..., uppercase letters are entered as `\Gamma`, `\Delta`, ...[2]

```
$\lambda,\xi,\pi,\mu,\Phi,\Omega$
```

$\lambda, \xi, \pi, \mu, \Phi, \Omega$

Exponents and Subscripts can be specified using the ^ and the _ character.

```
$a_{1}$ \qquad $x^{2}$ \qquad
$e^{-\alpha t}$ \qquad
$a^{3}_{ij}$\\
$e^{x^2} \neq {e^x}^2$
```

$a_1 \qquad x^2 \qquad e^{-\alpha t} \qquad a^3_{ij}$
$e^{x^2} \neq {e^x}^2$

[2]There is no uppercase Alpha defined in LaTeX 2ε because it looks the same as a normal roman A. Once the new math coding is done, things will change.

The **square root** is entered as `\sqrt`, the n^{th} root is generated with `\sqrt[`*n*`]`. The size of the root sign is determined automatically by LaTeX. If just the sign is needed, use `\surd`.

```
$\sqrt{x}$ \qquad
$\sqrt{ x^{2}+\sqrt{y} }$
\qquad $\sqrt[3]{2}$\\[3pt]
$\surd[x^2 + y^2]$
```

$\sqrt{x} \qquad \sqrt{x^2+\sqrt{y}} \qquad \sqrt[3]{2}$
$\surd[x^2 + y^2]$

The commands `\overline` and `\underline` create **horizontal lines** directly over or under an expression.

```
$\overline{m+n}$
```

$\overline{m+n}$

The commands `\overbrace` and `\underbrace` create long **horizontal braces** over or under an expression.

```
$\underbrace{ a+b+\cdots+z }_{26}$
```

$\underbrace{a+b+\cdots+z}_{26}$

To add mathematical accents such as small arrows or tilde signs to variables, you can use the commands given in **kd03**. Wide hats and tildes covering several characters are generated with `\widetilde` and `\widehat`. The `'` symbol gives a prime.

```
\begin{displaymath}
y=x^{2}\qquad y'=2x\qquad y''=2
\end{displaymath}
```

$$y = x^2 \qquad y' = 2x \qquad y'' = 2$$

Vectors often are specified by adding a small arrow symbol on top of a variable. This is done with the `\vec` command. The two commands `\overrightarrow` and `\overleftarrow` are useful to denote the vector from *A* to *B*.

```
\begin{displaymath}
\vec a\quad\overrightarrow{AB}
\end{displaymath}
```

$$\vec a \quad \overrightarrow{AB}$$

Names of log-like functions are often typeset in an upright font and not in italic like variables. Therefore LaTeX supplies the following commands to typeset the most important function names:

```
\arccos   \cos    \csc   \exp   \ker      \limsup  \min   \sinh
\arcsin   \cosh   \deg   \gcd   \lg       \ln      \Pr    \sup
\arctan   \cot    \det   \hom   \lim      \log     \sec   \tan
\arg      \coth   \dim   \inf   \liminf   \max     \sin   \tanh
```

```
\[\lim_{x \rightarrow 0}
\frac{\sin x}{x}=1\]
```

$$\lim_{x\rightarrow 0}\frac{\sin x}{x}=1$$

For the modulo function, there are two commands: `\bmod` for the binary operator "$a \bmod b$" and `\pmod` for expressions such as "$x \equiv a \pmod{b}$."

A built-up **fraction** is typeset with the `\frac{...}{...}` command. Often the slashed form $1/2$ is preferable, because it looks better for small amounts of 'fraction material.'

```
$1\frac{1}{2}$~hours
\begin{displaymath}
\frac{ x^{2} }{ k+1 }\qquad
x^{ \frac{2}{k+1} }\qquad
x^{ 1/2 }
\end{displaymath}
```

$1\frac{1}{2}$ hours

$$\frac{x^{2}}{k+1}\qquad x^{\frac{2}{k+1}}\qquad x^{1/2}$$

To typeset binomial coefficients or similar structures, you can use either the command `\binom`{*num*}{*denom*} or `\genfrac`{*ldelim*}{*rdelim*}{*thickness*}{*style*}{*num*}{*denom*}. The second command can be used to produce customized fraction like output and more information can be found in **mgbcr04**.

```
\begin{displaymath}
\binom{n}{k}\qquad
\genfrac{}{}{0pt}{}{x}{y+2}
\end{displaymath}
```

$$\binom{n}{k}\qquad\genfrac{}{}{0pt}{}{x}{y+2}$$

The **integral operator** is generated with `\int`, the **sum operator** with `\sum`. The upper and lower limits are specified with ^ and _ like subscripts and superscripts.

```
\begin{displaymath}
\sum_{i=1}^{n} \qquad
\int_{0}^{\frac{\pi}{2}} \qquad
\end{displaymath}
```

$$\sum_{i=1}^{n}\qquad\int_{0}^{\frac{\pi}{2}}\qquad$$

For **braces** and other delimiters, there exist all types of symbols in TEX (e.g. $[\ \langle\ \|\ \Updownarrow$). Round and square braces can be entered with the corresponding keys, curly braces with `\{`, all other delimiters are generated with special commands (e.g. `\updownarrow`). For a list of all delimiters available, check **kd03**.

```
\begin{displaymath}
{a,b,c}\neq\{a,b,c\}
\end{displaymath}
```

$${a,b,c}\neq\{a,b,c\}$$

If you put the command `\left` in front of an opening delimiter or `\right` in front of a closing delimiter, TEX will automatically determine the correct size of the delimiter. Note that you must close every `\left` with a corresponding

`\right`, and that the size is determined correctly only if both are typeset on the same line. If you don't want anything on the right, use the invisible '`\right .`'!

```
\begin{displaymath}
1 + \left( \frac{1}{ 1-x^{2} }
    \right) ^3
\end{displaymath}
```

$$1+\left(\frac{1}{1-x^{2}}\right)^{3}$$

In some cases it is necessary to specify the correct size of a mathematical delimiter by hand, which can be done using the commands `\big`, `\Big`, `\bigg` and `\Bigg` as prefixes to most delimiter commands.[3]

```
$\Big( (x+1) (x-1) \Big) ^{2}$\\
$\big(\Big(\bigg(\Bigg($\quad
$\big\}\Big\}\bigg\}\Bigg\}$\quad
$\big\|\Big\|\bigg\|\Bigg\|$
```

$\Big((x+1)(x-1)\Big)^{2}$
$\big(\Big(\bigg(\Bigg($ $\quad$ $\big\}\Big\}\bigg\}\Bigg\}$ $\quad$ $\big\|\Big\|\bigg\|\Bigg\|$

To enter **three dots** into a formula, you can use several commands. `\ldots` typesets the dots on the baseline, `\cdots` sets them centered. Besides that, there are the commands `\vdots` for vertical and `\ddots` for diagonal dots. You can find another example in section **??**.

```
\begin{displaymath}
x_{1},\ldots,x_{n} \qquad
x_{1}+\cdots+x_{n}
\end{displaymath}
```

$$x_{1},\ldots,x_{n} \qquad x_{1}+\cdots+x_{n}$$

A.4 Math Spacing

If the spaces within formulae chosen by TEX are not satisfactory, they can be adjusted by inserting special spacing commands. There are some commands for small spaces: `\,` for $\frac{3}{18}$ quad (␣), `\:` for $\frac{4}{18}$ quad (␣) and `\;` for $\frac{5}{18}$ quad (␣). The escaped space character `\␣` generates a medium sized space and `\quad` (␣) and `\qquad` (␣) produce large spaces. The size of a quad corresponds to the width of the character 'M' of the current font. The `\!` command produces a negative space of $-\frac{3}{18}$ quad (␣).

[3]These commands do not work as expected if a size changing command has been used, or the `11pt` or `12pt` option has been specified. Use the exscale or amsmath packages to correct this behaviour.

```
\newcommand{\rd}{\mathrm{d}}
\begin{displaymath}
\int\!\!\!\int_{D} g(x,y)
  \, \rd x\, \rd y
\end{displaymath}
instead of
\begin{displaymath}
\int\int_{D} g(x,y)\rd x \rd y
\end{displaymath}
```

$$\int\!\!\!\int_{D} g(x,y)\,\mathrm{d}x\,\mathrm{d}y$$

instead of

$$\int\int_{D} g(x,y)\mathrm{d}x\mathrm{d}y$$

Note that 'd' in the differential is conventionally set in roman.

AMS-LaTeX provides another way for fine tuning the spacing between multiple integral signs, namely the `\iint`, `\iiint`, `\iiiint`, and `\idotsint` commands. With the `amsmath` package loaded, the above example can be typeset this way:

```
\newcommand{\rd}{\mathrm{d}}
\begin{displaymath}
\iint_{D} \, \rd x \, \rd y
\end{displaymath}
```

$$\iint_{D}\,\mathrm{d}x\,\mathrm{d}y$$

See the electronic document testmath.tex (distributed with AMS-LaTeX) or Chapter 8 of "The LaTeX Companion"[4] for further details.

A.5 Vertically Aligned Material

To typeset **arrays**, use the `array` environment. It works somewhat similar to the `tabular` environment. The `\\` command is used to break the lines.

```
\begin{displaymath}
\mathbf{X} =
\left( \begin{array}{ccc}
x_{11} & x_{12} & \ldots \\
x_{21} & x_{22} & \ldots \\
\vdots & \vdots & \ddots
\end{array} \right)
\end{displaymath}
```

$$\mathbf{X} = \left(\begin{array}{ccc} x_{11} & x_{12} & \ldots \\ x_{21} & x_{22} & \ldots \\ \vdots & \vdots & \ddots \end{array} \right)$$

The `array` environment can also be used to typeset expressions which have one big delimiter by using a "`.`" as an invisible right delimiter:

[4]available at `CTAN:/tex-archive/info/ch8.*`.

```
\begin{displaymath}
y = \left\{ \begin{array}{ll}
 a & \textrm{if $d>c$}\\
 b+x & \textrm{in the morning}\\
 l & \textrm{all day long}
  \end{array} \right.
\end{displaymath}
```

$$y = \left\{ \begin{array}{ll} a & \textrm{if } d>c \\ b+x & \textrm{in the morning} \\ l & \textrm{all day long} \end{array} \right.$$

For formulae running over several lines or for equation systems, you can use the environments `eqnarray`, and `eqnarray*` instead of `equation`. In `eqnarray` each line gets an equation number. The `eqnarray*` does not number anything.

The `eqnarray` and the `eqnarray*` environments work like a 3-column table of the form `{rcl}`, where the middle column can be used for the equal sign or the not-equal sign. Or any other sign you see fit. The `\\` command breaks the lines.

```
\begin{eqnarray}
f(x)  & = & \cos x      \\
f'(x)  & = & -\sin x    \\
\int_{0}^{x} f(y)dy &
 = & \sin x
\end{eqnarray}
```

$$\begin{array}{rcll} f(x) & = & \cos x & \text{(A.5)} \\ f'(x) & = & -\sin x & \text{(A.6)} \\ \int_0^x f(y)dy & = & \sin x & \text{(A.7)} \end{array}$$

Notice that the space on either side of the the equal signs is rather large. It can be reduced by setting `\setlength\arraycolsep{2pt}`, as in the next example.

Long equations will not be automatically divided into neat bits. The author has to specify where to break them and how much to indent. The following two methods are the most common ones used to achieve this.

```
{\setlength\arraycolsep{2pt}
\begin{eqnarray}\notag
\sin x & = & x -\frac{x^{3}}{3!}
     +\frac{x^{5}}{5!}-{}
                 \\\notag
 & & {}-\frac{x^{7}}{7!}+{}\cdots
\end{eqnarray}}
```

$$\begin{array}{rcl} \sin x & = & x - \frac{x^3}{3!} + \frac{x^5}{5!} - \\ & & - \frac{x^7}{7!} + \cdots \end{array}$$

```
\begin{eqnarray}\notag
\lefteqn{ \cos x = 1
     -\frac{x^{2}}{2!} +{} }
                 \\\notag
 & & {}+\frac{x^{4}}{4!}
     -\frac{x^{6}}{6!}+{}\cdots
\end{eqnarray}
```

$$\begin{array}{l} \cos x = 1 - \frac{x^2}{2!} + \\ \qquad + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots \end{array}$$

The `\notag` command causes LaTeX to not generate a number for this equation.

It can be difficult to get vertically aligned equations to look right with these methods; the package amsmath provides a more powerful set of alternatives.

A.6 Math Font Size

In math mode, TEX selects the font size according to the context. Superscripts, for example, get typeset in a smaller font. If you want to typeset part of an equation in roman, don't use the `\textrm` command, because the font size switching mechanism will not work, as `\textrm` temporarily escapes to text mode. Use `\mathrm` instead to keep the size switching mechanism active. But pay attention, `\mathrm` will only work well on short items. Spaces are still not active and accented characters do not work.[5]

```
\begin{equation}
2^{\textrm{nd}} \quad
2^{\mathrm{nd}}
\end{equation}
```

$$2^{\textrm{nd}} \quad 2^{\mathrm{nd}} \tag{A.8}$$

Nevertheless, sometimes you need to tell LATEX the correct font size. In math mode, the font size is set with the four commands:

displaystyle (123), textstyle (123), scriptstyle ($\scriptstyle 123$) and scriptscriptstyle ($\scriptscriptstyle 123$).

Changing styles also affects the way limits are displayed.

```
\begin{displaymath}
\mathop{\mathrm{corr}}(X,Y)=
 \frac{\displaystyle
   \sum_{i=1}^n(x_i-\overline x)
   (y_i-\overline y)}
  {\displaystyle\biggl[
 \sum_{i=1}^n(x_i-\overline x)^2
\sum_{i=1}^n(y_i-\overline y)^2
\biggr]^{1/2}}
\end{displaymath}
```

$$\mathop{\mathrm{corr}}(X,Y)=\frac{\displaystyle\sum_{i=1}^n(x_i-\overline x)(y_i-\overline y)}{\displaystyle\biggl[\sum_{i=1}^n(x_i-\overline x)^2\sum_{i=1}^n(y_i-\overline y)^2\biggr]^{1/2}}$$

This is one of those examples in which we need larger brackets than the standard `\left[ \right]` provides.

A.7 Theorems, Laws, ...

When writing mathematical documents, you probably need a way to typeset "Lemmas", "Definitions", "Axioms" and similar structures. LATEX supports this with the command

newtheorem{*name*}[*counter*]{*text*}[*section*]

The *name* argument, is a short keyword used to identify the "theorem". With the *text* argument, you define the actual name of the "theorem" which will be printed in the final document.

[5]The AMS-LATEX package makes the textrm command work with size changing.

The arguments in square brackets are optional. They are both used to specify the numbering used on the “theorem”. With the *counter* argument you can specify the *name* of a previously declared “theorem”. The new “theorem” will then be numbered in the same sequence. The *section* argument allows you to specify the sectional unit within which you want your “theorem” to be numbered.

After executing the newtheorem command in the preamble of your document, you can use the following command within the document.

> `\begin{`*name*`}[`*text*`]`
>
> This is my interesting theorem
>
> `\end{`*name*`}`

This should be enough theory. The following examples will hopefully remove the final remains of doubt and make it clear that the `\newtheorem` environment is way too complex to understand.

```
% definitions for the document
% preamble
\newtheorem{law}{Law}
\newtheorem{jury}[law]{Jury}
%in the document
\begin{law} \label{law:box}
Don’t hide in the witness box
\end{law}
\begin{jury}[The Twelve]
It could be you! So beware and
see law~\ref{law:box}\end{jury}
\begin{law}No, No, No\end{law}
```

Law 1. *Don't hide in the witness box*

Jury 2 (The Twelve)**.** *It could be you! So beware and see law* **??**

Law 3. *No, No, No*

The “Jury” theorem uses the same counter as the “Law” theorem. Therefore it gets a number which is in sequence with the other “Laws”. The argument in square brackets is used to specify a title or something similar for the theorem.

```
\flushleft
\newtheorem{mur}{Murphy}[section]
\begin{mur}
If there are two or more
ways to do something, and
one of those ways can result
in a catastrophe, then
someone will do it.\end{mur}
```

Murphy A.7.1. *If there are two or more ways to do something, and one of those ways can result in a catastrophe, then someone will do it.*

The “Murphy” theorem gets a number which is linked to the number of the current section. You could also use another unit, for example chapter or subsection.

A.8 Bold symbols

It is quite difficult to get bold symbols in LaTeX; this is probably intentional as amateur typesetters tend to overuse them. The font change command `\mathbf` gives bold letters, but these are roman (upright) whereas mathematical symbols are normally italic. There is a `\boldmath` command, but *this can only be used outside mathematics mode*. It works for symbols too.

```
\begin{displaymath}\label{boldmath}
\mu, M \qquad \mathbf{M} \qquad
\mbox{\boldmath $\mu, M$}
\end{displaymath}
```

$$\mu, M \qquad \mathbf{M} \qquad \mu, M$$

Notice that the comma is bold too, which may not be what is required.

The package `amsbsy` (included by `amsmath`) makes this much easier as it includes a `\boldsymbol` command.

```
\begin{displaymath}\label{boldsymbol}
\mu, M \qquad
\boldsymbol{\mu}, \boldsymbol{M}
\end{displaymath}
```

$$\mu, M \qquad \mu, M$$

A.9 List of Mathematical Symbols

In the following tables, you find all the symbols normally accessible from *math mode*.

To use the symbols listed in Tables **??**–**??**,[6] the package `amssymb` must be loaded in the preamble of the document and the AMS math fonts must be installed, on the system. If the AMS package and fonts are not installed, on your system, have a look at

`CTAN:/tex-archive/macros/latex/required/amslatex`

Table A.1: Math Mode Accents.

$\hat{a}$	`\hat{a}`	$\check{a}$	`\check{a}`	$\tilde{a}$	`\tilde{a}`	$\acute{a}$	`\acute{a}`
$\grave{a}$	`\grave{a}`	$\dot{a}$	`\dot{a}`	$\ddot{a}$	`\ddot{a}`	$\breve{a}$	`\breve{a}`
$\bar{a}$	`\bar{a}`	$\vec{a}$	`\vec{a}`	$\widehat{A}$	`\widehat{A}`	$\widetilde{A}$	`\widetilde{A}`

[6]These tables were derived from `symbols.tex` by David Carlisle and subsequently changed extensively as suggested by Josef Tkadlec.

Table A.2: Lowercase Greek Letters.

α	`\alpha`	θ	`\theta`	o	`o`	υ	`\upsilon`
β	`\beta`	ϑ	`\vartheta`	π	`\pi`	ϕ	`\phi`
γ	`\gamma`	ι	`\iota`	ϖ	`\varpi`	φ	`\varphi`
δ	`\delta`	κ	`\kappa`	ρ	`\rho`	χ	`\chi`
ϵ	`\epsilon`	λ	`\lambda`	ϱ	`\varrho`	ψ	`\psi`
ε	`\varepsilon`	μ	`\mu`	σ	`\sigma`	ω	`\omega`
ζ	`\zeta`	ν	`\nu`	ς	`\varsigma`		
η	`\eta`	ξ	`\xi`	τ	`\tau`		

Table A.3: Uppercase Greek Letters.

Γ	`\Gamma`	Λ	`\Lambda`	Σ	`\Sigma`	Ψ	`\Psi`
Δ	`\Delta`	Ξ	`\Xi`	Υ	`\Upsilon`	Ω	`\Omega`
Θ	`\Theta`	Π	`\Pi`	Φ	`\Phi`		

Table A.4: Binary Relations.

You can produce corresponding negations by adding a `\not` command as prefix to the following symbols.

$<$	`<`	$>$	`>`	$=$	`=`
$\leq$	`\leq` or `\le`	$\geq$	`\geq` or `\ge`	$\equiv$	`\equiv`
$\ll$	`\ll`	$\gg$	`\gg`	$\doteq$	`\doteq`
$\prec$	`\prec`	$\succ$	`\succ`	$\sim$	`\sim`
$\preceq$	`\preceq`	$\succeq$	`\succeq`	$\simeq$	`\simeq`
$\subset$	`\subset`	$\supset$	`\supset`	$\approx$	`\approx`
$\subseteq$	`\subseteq`	$\supseteq$	`\supseteq`	$\cong$	`\cong`
$\sqsubset$	`\sqsubset` [a]	$\sqsupset$	`\sqsupset` [a]		`\Join` [a]
$\sqsubseteq$	`\sqsubseteq`	$\sqsupseteq$	`\sqsupseteq`	$\bowtie$	`\bowtie`
$\in$	`\in`	$\ni$	`\ni` , `\owns`	$\propto$	`\propto`
$\vdash$	`\vdash`	$\dashv$	`\dashv`	$\models$	`\models`
$\mid$	`\mid`	$\parallel$	`\parallel`	$\perp$	`\perp`
$\smile$	`\smile`	$\frown$	`\frown`	$\asymp$	`\asymp`
$:$	`:`	$\notin$	`\notin`	$\neq$	`\neq` or `\ne`

[a]Use the `latexsym` package to access this symbol

Table A.5: Binary Operators.

$+$	`+`	$-$	`-`		
$\pm$	`\pm`	$\mp$	`\mp`	$\triangleleft$	`\triangleleft`
$\cdot$	`\cdot`	$\div$	`\div`	$\triangleright$	`\triangleright`
$\times$	`\times`		`\setminus`	$\star$	`\star`
$\cup$	`\cup`	$\cap$	`\cap`	$\ast$	`\ast`
$\sqcup$	`\sqcup`	$\sqcap$	`\sqcap`	$\circ$	`\circ`
$\vee$	`\vee` , `\lor`	$\wedge$	`\wedge` , `\land`	$\bullet$	`\bullet`
$\oplus$	`\oplus`	$\ominus$	`\ominus`	$\diamond$	`\diamond`
$\odot$	`\odot`	$\oslash$	`\oslash`	$\uplus$	`\uplus`
$\otimes$	`\otimes`	$\bigcirc$	`\bigcirc`	$\amalg$	`\amalg`
$\bigtriangleup$	`\bigtriangleup`	$\bigtriangledown$	`\bigtriangledown`	$\dagger$	`\dagger`
$\lhd$	`\lhd` [a]	$\rhd$	`\rhd` [a]	$\ddagger$	`\ddagger`
$\unlhd$	`\unlhd` [a]	$\unrhd$	`\unrhd` [a]	$\wr$	`\wr`

Table A.6: BIG Operators.

$\sum$	`\sum`	$\bigcup$	`\bigcup`	$\bigvee$	`\bigvee`	$\bigoplus$	`\bigoplus`
$\prod$	`\prod`	$\bigcap$	`\bigcap`	$\bigwedge$	`\bigwedge`	$\bigotimes$	`\bigotimes`
$\coprod$	`\coprod`	$\bigsqcup$	`\bigsqcup`			$\bigodot$	`\bigodot`
$\int$	`\int`	$\oint$	`\oint`			$\biguplus$	`\biguplus`

Table A.7: Arrows.

$\leftarrow$	`\leftarrow` or `\gets`	$\longleftarrow$	`\longleftarrow`	$\uparrow$	`\uparrow`
$\rightarrow$	`\rightarrow` or `\to`	$\longrightarrow$	`\longrightarrow`	$\downarrow$	`\downarrow`
$\leftrightarrow$	`\leftrightarrow`	$\longleftrightarrow$	`\longleftrightarrow`	$\updownarrow$	`\updownarrow`
$\Leftarrow$	`\Leftarrow`	$\Longleftarrow$	`\Longleftarrow`	$\Uparrow$	`\Uparrow`
$\Rightarrow$	`\Rightarrow`	$\Longrightarrow$	`\Longrightarrow`	$\Downarrow$	`\Downarrow`
$\Leftrightarrow$	`\Leftrightarrow`	$\Longleftrightarrow$	`\Longleftrightarrow`	$\Updownarrow$	`\Updownarrow`
$\mapsto$	`\mapsto`	$\longmapsto$	`\longmapsto`	$\nearrow$	`\nearrow`
$\hookleftarrow$	`\hookleftarrow`	$\hookrightarrow$	`\hookrightarrow`	$\searrow$	`\searrow`
$\leftharpoonup$	`\leftharpoonup`	$\rightharpoonup$	`\rightharpoonup`	$\swarrow$	`\swarrow`
$\leftharpoondown$	`\leftharpoondown`	$\rightharpoondown$	`\rightharpoondown`	$\nwarrow$	`\nwarrow`
$\rightleftharpoons$	`\rightleftharpoons`	$\iff$	`\iff` (bigger spaces)	$\leadsto$	`\leadsto` [a]

[a]Use the `latexsym` package to access this symbol

Table A.8: Delimiters.

$($	`(`	$)$	`)`	$\uparrow$	`\uparrow`	$\Uparrow$	`\Uparrow`
$[$	`[` or `\lbrack`	$]$	`]` or `\rbrack`	$\downarrow$	`\downarrow`	$\Downarrow$	`\Downarrow`
$\{$	`\{` or `\lbrace`	$\}$	`\}` or `\rbrace`	$\updownarrow$	`\updownarrow`	$\Updownarrow$	`\Updownarrow`
$\langle$	`\langle`	$\rangle$	`\rangle`	$\vert$	`|` or `\vert`	$\Vert$	`\\|` or `\Vert`
$\lfloor$	`\lfloor`	$\rfloor$	`\rfloor`	$\lceil$	`\lceil`	$\rceil$	`\rceil`
$/$	`/`	$\backslash$	`\backslash`		. (dual. empty)		

Table A.9: Large Delimiters.

$\lgroup$	`\lgroup`	$\rgroup$	`\rgroup`		`\lmoustache`		`\rmoustache`
	`\arrowvert`		`\Arrowvert`		`\bracevert`		

Table A.10: Miscellaneous Symbols.

$\dots$	`\dots`	$\cdots$	`\cdots`	$\vdots$	`\vdots`	$\ddots$	`\ddots`
$\hbar$	`\hbar`	$\imath$	`\imath`	$\jmath$	`\jmath`	ℓ	`\ell`
$\Re$	`\Re`	$\Im$	`\Im`	$\aleph$	`\aleph`	$\wp$	`\wp`
$\forall$	`\forall`	$\exists$	`\exists`	$\mho$	`\mho` [a]	∂	`\partial`
$'$	`'`	$\prime$	`\prime`	$\emptyset$	`\emptyset`	∞	`\infty`
∇	`\nabla`	$\triangle$	`\triangle`	$\Box$	`\Box` [a]	$\Diamond$	`\Diamond` [a]
$\bot$	`\bot`	$\top$	`\top`	$\angle$	`\angle`	$\surd$	`\surd`
$\diamondsuit$	`\diamondsuit`	$\heartsuit$	`\heartsuit`	$\clubsuit$	`\clubsuit`	$\spadesuit$	`\spadesuit`
$\neg$	`\neg` or `\lnot`	$\flat$	`\flat`	$\natural$	`\natural`	$\sharp$	`\sharp`

[a]Use the `latexsym` package to access this symbol

Table A.11: Non-Mathematical Symbols.

These symbols can also be used in text mode.

†	`\dag`	§	`\S`	©	`\copyright`
‡	`\ddag`	¶	`\P`	£	`\pounds`

Table A.12: AMS Delimiters.

$\ulcorner$	`\ulcorner`	$\urcorner$	`\urcorner`	$\llcorner$	`\llcorner`	$\lrcorner$	`\lrcorner`

Table A.13: AMS Greek and Hebrew.

	`\digamma`	$\varkappa$	`\varkappa`	$\beth$	`\beth`	$\daleth$	`\daleth`	$\gimel$	`\gimel`

Table A.14: AMS Binary Relations.

$\lessdot$	`\lessdot`	$\gtrdot$	`\gtrdot`	$\doteqdot$	`\doteqdot` or `\Doteq`
$\leqslant$	`\leqslant`	$\geqslant$	`\geqslant`	$\risingdotseq$	`\risingdotseq`
$\eqslantless$	`\eqslantless`	$\eqslantgtr$	`\eqslantgtr`	$\fallingdotseq$	`\fallingdotseq`
$\leqq$	`\leqq`	$\geqq$	`\geqq`	$\eqcirc$	`\eqcirc`
$\lll$	`\lll` or `\llless`	$\ggg$	`\ggg` or `\gggtr`	$\circeq$	`\circeq`
$\lesssim$	`\lesssim`	$\gtrsim$	`\gtrsim`	$\triangleq$	`\triangleq`
$\lessapprox$	`\lessapprox`	$\gtrapprox$	`\gtrapprox`	$\bumpeq$	`\bumpeq`
$\lessgtr$	`\lessgtr`	$\gtrless$	`\gtrless`	$\Bumpeq$	`\Bumpeq`
$\lesseqgtr$	`\lesseqgtr`	$\gtreqless$	`\gtreqless`	$\thicksim$	`\thicksim`
$\lesseqqgtr$	`\lesseqqgtr`	$\gtreqqless$	`\gtreqqless`	$\thickapprox$	`\thickapprox`
$\preccurlyeq$	`\preccurlyeq`	$\succcurlyeq$	`\succcurlyeq`	$\approxeq$	`\approxeq`

Table A.15: AMS Binary Relations Continued.

$\curlyeqprec$	`\curlyeqprec`	$\curlyeqsucc$	`\curlyeqsucc`	$\backsim$	`\backsim`
$\precsim$	`\precsim`	$\succsim$	`\succsim`	$\backsimeq$	`\backsimeq`
	`\precapprox`		`\succapprox`	$\vDash$	`\vDash`
	`\subseteqq`		`\supseteqq`	$\Vdash$	`\Vdash`
$\Subset$	`\Subset`	$\Supset$	`\Supset`	$\Vvdash$	`\Vvdash`
$\sqsubset$	`\sqsubset`	$\sqsupset$	`\sqsupset`		`\backepsilon`
$\therefore$	`\therefore`	$\because$	`\because`	$\varpropto$	`\varpropto`
$\shortmid$	`\shortmid`	$\shortparallel$	`\shortparallel`	$\between$	`\between`
$\smallsmile$	`\smallsmile`	$\smallfrown$	`\smallfrown`		`\pitchfork`
$\vartriangleleft$	`\vartriangleleft`	$\vartriangleright$	`\vartriangleright`	$\blacktriangleleft$	`\blacktriangleleft`
$\trianglelefteq$	`\trianglelefteq`	$\trianglerighteq$	`\trianglerighteq`	$\blacktriangleright$	`\blacktriangleright`

Table A.16: AMS Arrows.

$\dashleftarrow$	`\dashleftarrow`	$\dashrightarrow$	`\dashrightarrow`	$\multimap$	`\multimap`
$\leftleftarrows$	`\leftleftarrows`	$\rightrightarrows$	`\rightrightarrows`	$\upuparrows$	`\upuparrows`
$\leftrightarrows$	`\leftrightarrows`	$\rightleftarrows$	`\rightleftarrows`	$\downdownarrows$	`\downdownarrows`
$\Lleftarrow$	`\Lleftarrow`	$\Rrightarrow$	`\Rrightarrow`	$\upharpoonleft$	`\upharpoonleft`
$\twoheadleftarrow$	`\twoheadleftarrow`	$\twoheadrightarrow$	`\twoheadrightarrow`	$\upharpoonright$	`\upharpoonright`
$\leftarrowtail$	`\leftarrowtail`	$\rightarrowtail$	`\rightarrowtail`	$\downharpoonleft$	`\downharpoonleft`
$\leftrightharpoons$	`\leftrightharpoons`	$\rightleftharpoons$	`\rightleftharpoons`	$\downharpoonright$	`\downharpoonright`
$\Lsh$	`\Lsh`	$\Rsh$	`\Rsh`	$\rightsquigarrow$	`\rightsquigarrow`
$\looparrowleft$	`\looparrowleft`	$\looparrowright$	`\looparrowright`	$\leftrightsquigarrow$	`\leftrightsquigarrow`
$\curvearrowleft$	`\curvearrowleft`	$\curvearrowright$	`\curvearrowright`		
$\circlearrowleft$	`\circlearrowleft`	$\circlearrowright$	`\circlearrowright`		

Table A.17: AMS Negated Binary Relations and Arrows.

$\nless$	`\nless`	$\ngtr$	`\ngtr`	$\varsubsetneqq$	`\varsubsetneqq`
$\lneq$	`\lneq`	$\gneq$	`\gneq`	$\varsupsetneqq$	`\varsupsetneqq`
$\nleq$	`\nleq`	$\ngeq$	`\ngeq`	$\nsubseteqq$	`\nsubseteqq`
$\nleqslant$	`\nleqslant`	$\ngeqslant$	`\ngeqslant`	$\nsupseteqq$	`\nsupseteqq`
$\lneqq$	`\lneqq`	$\gneqq$	`\gneqq`	$\nmid$	`\nmid`
$\lvertneqq$	`\lvertneqq`	$\gvertneqq$	`\gvertneqq`	$\nparallel$	`\nparallel`
$\nleqq$	`\nleqq`	$\ngeqq$	`\ngeqq`	$\nshortmid$	`\nshortmid`
$\lnsim$	`\lnsim`	$\gnsim$	`\gnsim`	$\nshortparallel$	`\nshortparallel`
$\lnapprox$	`\lnapprox`	$\gnapprox$	`\gnapprox`	$\nsim$	`\nsim`
$\nprec$	`\nprec`	$\nsucc$	`\nsucc`	$\ncong$	`\ncong`
$\npreceq$	`\npreceq`	$\nsucceq$	`\nsucceq`	$\nvdash$	`\nvdash`
	`\precneqq`		`\succneqq`	$\nvDash$	`\nvDash`
$\precnsim$	`\precnsim`	$\succnsim$	`\succnsim`	$\nVdash$	`\nVdash`
	`\precnapprox`		`\succnapprox`	$\nVDash$	`\nVDash`
$\subsetneq$	`\subsetneq`	$\supsetneq$	`\supsetneq`	$\ntriangleleft$	`\ntriangleleft`
$\varsubsetneq$	`\varsubsetneq`	$\varsupsetneq$	`\varsupsetneq`	$\ntriangleright$	`\ntriangleright`
$\nsubseteq$	`\nsubseteq`	$\nsupseteq$	`\nsupseteq`	$\ntrianglelefteq$	`\ntrianglelefteq`
	`\subsetneqq`		`\supsetneqq`	$\ntrianglerighteq$	`\ntrianglerighteq`
$\nleftarrow$	`\nleftarrow`	$\nrightarrow$	`\nrightarrow`	$\nleftrightarrow$	`\nleftrightarrow`
$\nLeftarrow$	`\nLeftarrow`	$\nRightarrow$	`\nRightarrow`	$\nLeftrightarrow$	`\nLeftrightarrow`

Table A.18: AMS Binary Operators.

$\dotplus$	`\dotplus`	$\centerdot$	`\centerdot`	$\intercal$	`\intercal`
$\ltimes$	`\ltimes`	$\rtimes$	`\rtimes`	$\divideontimes$	`\divideontimes`
$\Cup$	`\Cup` or `\doublecup`	$\Cap$	`\Cap` or `\doublecap`	$\smallsetminus$	`\smallsetminus`
$\veebar$	`\veebar`	$\barwedge$	`\barwedge`		`\doublebarwedge`
$\boxplus$	`\boxplus`	$\boxminus$	`\boxminus`	$\circleddash$	`\circleddash`
$\boxtimes$	`\boxtimes`	$\boxdot$	`\boxdot`	$\circledcirc$	`\circledcirc`
$\leftthreetimes$	`\leftthreetimes`	$\rightthreetimes$	`\rightthreetimes`	$\circledast$	`\circledast`
$\curlyvee$	`\curlyvee`	$\curlywedge$	`\curlywedge`		

Table A.19: AMS Miscellaneous.

$\hbar$	`\hbar`	$\hslash$	`\hslash`	$\Bbbk$	`\Bbbk`
$\square$	`\square`	$\blacksquare$	`\blacksquare`	$\circledS$	`\circledS`
	`\vartriangle`		`\blacktriangle`	$\complement$	`\complement`
	`\triangledown`		`\blacktriangledown`		`\Game`
$\lozenge$	`\lozenge`	$\blacklozenge$	`\blacklozenge`		`\bigstar`
$\angle$	`\angle`	$\measuredangle$	`\measuredangle`	$\sphericalangle$	`\sphericalangle`
$\diagup$	`\diagup`	$\diagdown$	`\diagdown`	$\backprime$	`\backprime`
$\nexists$	`\nexists`		`\Finv`	$\varnothing$	`\varnothing`
$\eth$	`\eth`	$\mho$	`\mho`		

Table A.20: Math Alphabets.

Example	Command	Required package
ABCdef	`\mathrm{ABCdef}`	
ABCdef	`\mathit{ABCdef}`	
$ABCdef$	`\mathnormal{ABCdef}`	
$\mathcal{ABC}$	`\mathcal{ABC}`	
$\mathcal{ABC}$	`\mathcal{ABC}`	`eucal` with option: or
	`\mathscr{ABC}`	`eucal` with option: mathscr
$\mathfrak{ABCdef}$	`\mathfrak{ABCdef}`	`eufrak`
$\mathbb{ABC}$	`\mathbb{ABC}`	`amsfonts` or `amssymb`

APPENDIX B

Examples of Java Code

Here are some examples of Java source using the `listings` package. I have entered the following before any code examples to format the code as shown.

```
\lstset{language=java}
\lstset{backgroundcolor=\color{white},rulecolor=\color{black}}
\lstset{linewidth=.95\textwidth,breaklines=true}
\lstset{commentstyle=\textit,stringstyle=\upshape,showspaces=false}
\lstset{frame = trbl, frameround=tttt}
\lstset{numbers=left,numberstyle=\tiny,basicstyle=\small}
\lstset{commentstyle=\normalfont\itshape,breakautoindent=true}
\lstset{abovecaptionskip=1.2\baselineskip,xleftmargin=30pt}
\lstset{framesep=6pt}
```

I have included the code by entering

```
\begin{singlespace}
\lstinputlisting[caption=Clock Code,label=clock]{source/Clock.java}
\end{singlespace}
```

Listing B.1: Clock Code

```java
//file: Clock.java
public class Clock extends UpdateApplet {
    public void paint( java.awt.Graphics g ) {
        g.drawString( new java.util.Date().toString(  ), 10, 25 );
    }
}
```

Listing B.2: Consumer

```
//file: Consumer.java
import java.util.Vector;

public class Consumer implements Runnable
{
    Producer producer;

    Consumer( Producer producer ) {
        this.producer = producer;
    }

    public void run() {
        while ( true ) {
            String message = producer.getMessage();
            System.out.println("Got message: "+ message);
                        try {
                                Thread.sleep( 2000 );
                        } catch ( InterruptedException e ) { }
        }
    }

    public static void main(String args[]) {
        Producer producer = new Producer();
        new Thread( producer ).start();
        Consumer consumer = new Consumer( producer );
        new Thread( consumer ).start();
    }
}
```

Listing B.3: EvilEmpire Code

```
//file: EvilEmpire.java
import java.net.*;

public class EvilEmpire {
  public static void main(String[] args) throws Exception{
    try {
      Socket s = new Socket("???.???.???.???", 80);
      System.out.println("Connected!");
    }
    catch (SecurityException e) {
      System.out.println("SecurityException: could not connect.");
    }
  }
}
```

APPENDIX C

C++ Examples

This appendix demonstrates the `listings` package's ability to format C++ code.

Listing C.1: Motion Class

```
#include "Motion.h"


Motion::Motion(int _steps) : TimeSeries(_steps) {}

Motion::Motion(Noise2 *_noise) : TimeSeries(_noise->GetSteps()) {
  noise = _noise;
}

Motion::~Motion() {
  delete noise;
}

void Motion::SyncWithNoise() {
  if (noise != NULL) {
    this->Initialize();
    double sum = 0;
    int getsteps = this->GetSteps();
    for (int i = 0; i < getsteps; i++) {
      sum += noise->GetData(i);
      this->SetData(i, sum);
    }
  } else {
    fprintf(stderr, "%s\n", MOTION_NOISE_ERR);
  }
}
```

Listing C.2: Plotter Class

```
#include <unistd.h>
#include "Plotter.h"


void Plotter::MakePlot(char *filename) {
  ofstream fout(FILE_PLOT);
  fout << "set␣data␣style␣linespoints" << endl
       << "plot␣\"" << filename << "\"" << endl;
  fout.close();

  int pid, status;
  pid = fork();
  if (pid >= 0) {
    if (pid == 0) {
      execl(FILE_GNUPLOT, "gnuplot", "-persist", FILE_PLOT, NULL);
      fprintf(stderr, "%s␣\"gnuplot\"", EXEC_ERR);
      exit(0);
    } else {
      wait(status);
    }
  } else {
    fprintf(stderr, "%s␣\"gnuplot\"", FORK_ERR);
  }

  /*  pid = fork();
  if (pid >= 0) {
    if (pid == 0) {
      execlp("rm", FILE_PLOT, NULL);
      fprintf(stderr, "%s \"rm\"", EXEC_ERR);
      exit(0);
    } else {
      wait(status);
    }
  } else {
    fprintf(stderr, "%s \"rm\"", FORK_ERR);
    }*/

}
```

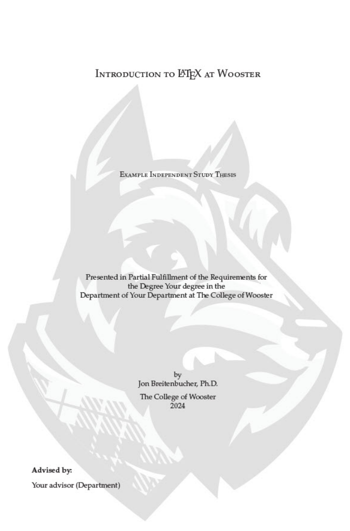
Listing C.3: Simulation Class

```
#include "Simulation.h"


Simulation::Simulation(int _steps, double H) {
  noise = new Noise2(_steps);
  motion = new Motion(noise);
}

Simulation::~Simulation() {
  delete noise;
  delete motion;
}

void Simulation::Analyze() {
  noiseplotter.MakePlot("noise");
  motionplotter.MakePlot("motion");
}
```

APPENDIX D

Cover and Copyright Examples

Here are images showing what the options `scottie`, `wblack`, and `woostercopyright` produce when used for typesetting an IS.



INTRODUCTION TO LaTeX AT WOOSTER

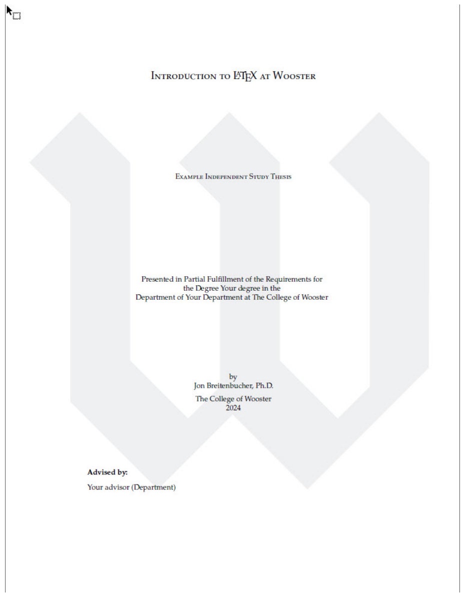
EXAMPLE INDEPENDENT STUDY THESIS

Presented in Partial Fulfillment of the Requirements for
the Degree Your degree in the
Department of Your Department at The College of Wooster

by
Jon Breitenbucher, Ph.D.
The College of Wooster
2024

Advised by:
Your advisor (Department)

(a) scottie option



INTRODUCTION TO LaTeX AT WOOSTER

EXAMPLE INDEPENDENT STUDY THESIS

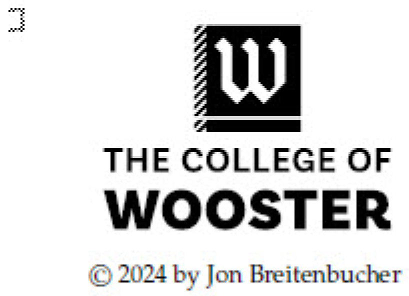
Presented in Partial Fulfillment of the Requirements for
the Degree Your degree in the
Department of Your Department at The College of Wooster

by
Jon Breitenbucher, Ph.D.
The College of Wooster
2024

Advised by:
Your advisor (Department)

(b) wblack option

Figure D.1: Options for the IS Cover Page


THE COLLEGE OF WOOSTER

© 2024 by Jon Breitenbucher

Figure D.2: woostercopytight option

Afterword

So how does a LaTeX session work? LaTeX loads the document class with any specified options and uses the information in the document class to decide on how the document will be formatted. At this point LaTeX loads any packages that the user has specified. Packages extend the basic LaTeX commands and formatting for special situations. `woosterthesis` loads several packages by default and several others through class options; it is assumed you have these installed on your system. They are: `alltt`, `amsfonts`, `amsmath`, `amssymb`, `amsthm`, `babel`, `biblatex`, `biblatex-chicago`, `caption`, `csquotes`, `eso-pic`, `eucal`, `eufrak`, `fancyhdr`, `float`, `floatflt`, `fontenc`, `fontspec`, `geometry`, `graphicx`, `hyperref`, `ifpdf`, `ifthen`, `ifxetex`, `inputenc`, `lettrine`, `listings`, `lmodern`, `makeidx`, `maple2e`, `microtype`, `pdftex`, `polyglossia`, `pxfonts`, `setspace`, `subfig`, `textpos`, `TikZ`, `verbatim`, `wrapfig`, `xcolor`, `xltxtra`, and `xunicode`. The `woosterthesis` class assumes you are using pdfTeX(support for postscript based TeX has been dropped as of 2006/17/11).

The `hyperref` package will make your thesis a linked document. `amsthm` is for altering the Theorem environments. `amsmath` implements almost all the mathematical symbols. `amssymb` adds the mathematical symbols not present in `amsmath`. `graphicx` and `eso-pic` are used to place graphics files in the thesis. `geometry` is used to set up the margins for the thesis. `setspace` is used to alter spacing by allowing a `singlespace`, `doublespace`, and `onehalfspace` environments. `biblatex` formats citations and references. Documentation is included for some of the packages in the `doc` folder.

These packages should all be installed with a full installation of TeXLive on OS X or Windows. On OS X one can use the the MacTeX installer as i-Installer is no longer supported as of 2007/1/1. On Windows one can use MikTeX to install all available packages which will install all the above. By default the MikTeX install does a minimal installation. You will need to run the updater to make your MikTeX installation aware of all the new packages.

There is also a new TeX engine called XeTeX which allows one to use the native fonts on your system as text fonts in the document. More information can be found at the XeTeX homepage. If using XeTeX you will also need `fontspec`, `xunicode`, and `xltxtra` which should be installed with XeTeX . The default font definitions for using XeTeX are in `styles/packages.tex` with suggestions for alternatives to the defaults that have been set.

Once the packages are loaded, LaTeX begins to process the commands contained between the `document` tags. As it processes the commands, several auxiliary files are created. These files contain information needed for things like the Bibliography, Table of Contents, List of Figures, etc. We then process the file a second time to allow LaTeX to use its auxiliary files to fill in information. Some information may require three passes before it is displayed. Once LaTeX is done you are presented with a PDF of the output.