

Luca Merciadri

# A Practical Guide to L<sup>A</sup>T<sub>E</sub>X Tips and Tricks

October 7, 2011

This page intentionally left blank.

To all L<sup>A</sup>T<sub>E</sub>X lovers who gave me the opportunity to learn a new way of not  
only *writing* things, but *thinking* them . . . Claudio Beccari, Karl Berry,  
David Carlisle, Robin Fairbairns, Enrico Gregorio, Stefan Kottwitz, Frank  
Mittelbach, Martin Münch, Heiko Oberdiek, Chris Rowley, Marc van  
Dongen, Joseph Wright, . . .

This page intentionally left blank.

---

# Contents

---

## Part I Standard Documents

---

<b>1</b>	<b>Major Tricks</b>	7
1.1	Allowing	10
1.1.1	Linebreaks After Comma in Math Mode	10
1.2	Avoiding	11
1.2.1	Erroneous Logic Formulae	11
1.2.2	Erroneous References for Floats	12
1.3	Counting	14
1.3.1	Introduction	14
1.3.2	Equations For an Appendix	16
1.3.3	Examples	16
1.3.4	Rows In Tables	16
1.4	Creating	17
1.4.1	Counters	17
1.4.2	Enumerate Lists With a Star	17
1.4.3	Math Math Operators	18
1.4.4	Math Operators	19
1.4.5	New Abstract Environments	20
1.4.6	Quotation Marks Using ”	21
1.4.7	Titlepages	22
1.5	Drawing	24
1.5.1	A Block Quote With Quotation Marks	24
1.5.2	A Box On Every Digit Of an Integer	25
1.5.3	A Centered Slash on Math Symbols	26
1.5.4	A Colored Front Cover	27
1.5.5	A (Single/Double) Linked List	28
1.5.6	A Personalized <code>overline</code>	29
1.5.7	Boxes In <code>align</code> -like Environments	30
1.5.8	Cases With Square Brackets	31
1.5.9	Closed Square Roots	32

1.5.10	Correct Circuits	32
1.5.11	Dependency Arrows	34
1.5.12	Diagonally-Cut Cells	34
1.5.13	Graph Paper	34
1.5.14	Logic Gates	37
1.5.15	Over and Under Braces on Same Elements	42
1.5.16	Rightmost Braces	43
1.5.17	Tables With Dashed Lines, Shaded Cells and Arrows Pointing At a Row and a Column	44
1.6	Finding	45
1.6.1	Symbols' Commands	45
1.7	Fixing Errors	45
1.7.1	Dimen	45
1.7.2	Fncychap	46
1.7.3	L <sup>A</sup> T <sub>E</sub> X	46
1.7.4	PDF compatibility	47
1.8	Incorporating	47
1.8.1	MATLAB Graphics	47
1.8.2	Spreadsheets	50
1.8.3	References	50
1.9	Modifying	50
1.9.1	Captions	50
1.9.2	Margins and Marginpar Notes	50
1.9.3	The Zoom of a Document	52
1.10	Printing	52
1.10.1	Empty Pages	52
1.10.2	Monochrome	52
1.11	Sizing	53
1.11.1	Boxed Minipages Depending On the Text	53
1.11.2	\left and \right to Produce Smaller Brackets	53
1.12	Splitting	54
1.12.1	algorithm Environments on Multiple Pages	54
1.13	Using	55
1.13.1	\centering and the center Environment Accordingly	55
1.13.2	\scalebox	55
1.13.3	The Enumerate Package	56
1.14	Writing	56
1.14.1	Aligned Systems of (In)Equations	56
1.14.2	(a) Bold \ell	57
1.14.3	A Symmetric Matrix	57
1.14.4	Code Depending On the Processor	58
1.14.5	Dancing Text	59
1.14.6	Dash Integrals	60
1.14.7	Dashed And Underlined Text	60
1.14.8	Date and Time	61

1.14.9	Diagonal (Inverse) Dots	61
1.14.10	Different Slides	62
1.14.11	Enumerations With Textcircled Numbers	63
1.14.12	Footnotes In Boxes	63
1.14.13	Footnotes In <code>tabular</code>	65
1.14.14	Footnotes With a New Number At Each Section	65
1.14.15	Footnotes Without Any Number	66
1.14.16	Fractions	66
1.14.17	Indented Text, But Not the First Line	66
1.14.18	Integrals Adapted to the Context	67
1.14.19	Links	70
1.14.20	Logic Inferences	71
1.14.21	Matrices	72
1.14.22	Minitocs	78
1.14.23	Product Integrals	79
1.14.24	Roman Numbers	79
1.14.25	Simplifications	80
1.14.26	Small <code>mathcal</code> Letters	80
1.14.27	Stacks (In Equations)	81
1.14.28	Standard Display of Date and Time	82
1.14.29	Subequations with Braces	82
1.14.30	Tables	83
1.14.31	Tabular Packages: What are Their Respective Purposes, and Which Ones Conflict?	83
1.14.32	Text On the Same Line as an Equation	85
1.14.33	Text Below an Image	86
1.14.34	Two Itemizations on the Same Line	86
1.14.35	Verbatim in Center	87
1.14.36	Verbatim In <code>tabular</code>	87
1.14.37	Verbatim In Footnotes	87
1.14.38	Verbatim In Hierarchy	87
1.14.39	Vertically Aligned Itemize Environments	88
<b>2</b>	<b>Fine Tuning</b>	89
2.1	Avoiding	90
2.1.1	Color In TOC And TOFs	90
2.2	Counting	91
2.2.1	Pages And Tables	91
2.2.2	Paragraphs	92
2.3	Defining	92
2.3.1	A Turning Page Text	92
2.3.2	Bold Index Numbers	93
2.3.3	Useful Commands	93
2.4	Managing	94
2.4.1	Beamer Presentations and Articles	94

2.4.2	Big Documents	96
2.4.3	Floats	97
2.4.4	Indentation	99
2.4.5	Index	99
2.5	Modifying	99
2.5.1	QED Square	99
2.6	Using	101
2.6.1	Cleardoublepage Command Correctly	101
2.6.2	Crops	101
2.6.3	Microtype	101
2.6.4	Pedagogical Tools	102
2.6.5	Units From the SI	102
2.7	Writing	103
2.7.1	Comments In an Effective Way	103
2.7.2	Derivatives and Such Symbols In a Better Way	103
2.7.3	Letters With the <code>lettre</code> Documentclass Without a Big Space	104
2.7.4	Letters With the <code>lettre</code> Documentclass Without the Line Bending	104
2.7.5	MATLAB Code	105
2.7.6	Messages on Would-Be Blank Pages	106
2.7.7	Unhabitual Punctuation Marks	107
2.8	Referring to <code>\labels</code> by long names	108
2.8.1	Code	108
2.8.2	Example	109
2.9	Personalizing The Index	109
2.9.1	Standard Personalization	109
2.9.2	French Tricks	109
2.9.3	Insensitive Letter Sort	110
2.9.4	Special Letter Sort	110
2.9.5	German Letter Sort	110
2.9.6	Math Formulae Sort	111
2.9.7	Greek Letter Sort	112
2.9.8	Special Characters Sort	113
2.9.9	Last Refinements	113
<b>3</b>	<b>Various Tricks</b>	115
3.1	Automatically Capitalizing First Letters of Words	115
3.2	Changing “References” to Some Other Text	115
3.3	Coloring the $\text{\LaTeX} 2_{\epsilon}$ Logo	115
3.3.1	Code	116
3.3.2	Example	116
3.4	Correct Spacing Around Square Brackets In Math Mode	117
3.5	Hiding the Column Separator in <code>multicol</code>	117

3.6	How Do You Choose Between Declaring a New Command Or a New Environment?	118
3.7	Naming Different Rows or Columns Of a Matrix Using Braces	118
3.7.1	Code	119
3.7.2	Example	120
3.8	Overfull <code>\hboxes</code>	120
3.9	Using the $\mathbb{I}$ Symbol for Maths	120
3.9.1	$\mathbb{I}$ in Signal Processing	121
3.9.2	$\mathbb{I}$ in $\text{\LaTeX}$	122
3.10	‘Why is $\text{\LaTeX}$ Separated Into Many Thousands of Packages?’	123
4	<b>A Special Case Study: The Springer <code>svmono</code> Class</b>	125
4.1	Introduction	125
4.2	The Problem Of <code>Tocdepth</code>	126

---

## Part II Special Documents

---

5	<b>AutoMath</b>	131
5.1	Generating Equations	131
5.1.1	Linear	131
5.1.2	Square	132
5.2	Generating Square Roots	133
5.2.1	Example	133
5.3	Generating Systems (Of Linear Equations)	133
5.3.1	Two Equations, Two Variables	133
5.3.2	Three Equations, Three Variables	135
6	<b>Envelopes</b>	137
6.1	C6 Standard And Adaptations	137
6.1.1	In the Printer	138
6.1.2	Output	138
6.2	Standards	139
7	<b>Letters</b>	141
7.1	English	141
7.2	French	142
8	<b>Posters</b>	145
8.1	With Some Color	145
8.1.1	Using Beamer	145
8.1.2	Notes on Other Approaches	147

<b>9</b>	<b>Watermarking</b>	149
9.1	The Current Document	149
9.1.1	The <code>background</code> Package	149
9.1.2	The <code>xwatermark</code> Option	150
9.1.3	The <code>TikZ</code> Way	150
9.1.4	The <code>draftcopy</code> package	150
9.2	An Included PDF Document	151
<b>10</b>	<b>Encoding Problems</b>	153
10.1	The <code>inputenc</code> Package: the Encoding of the Document	153
10.1.1	Description	153
10.1.2	Encoding Choice	153
10.1.3	Solutions?	154
10.1.4	Platforms	154
10.2	The Proper Encoding of the Document File	155
10.2.1	Directly Writing Characters Without Commands	155
10.2.2	Converting a File to the Good Encoding	156
10.3	Dealing With BiB Files	157
10.4	Font Encodings	158
10.5	Mapping Multiple Encodings Into One	158
10.6	Summary	159
<b>11</b>	<b><math>\LaTeX</math> 2<math>\epsilon</math> and Plagiarism</b>	161
<b>12</b>	<b>Turing Completeness</b>	167
<b>13</b>	<b>Con<math>\TeX</math>t, Lua<math>\TeX</math>, te<math>\TeX</math>, Xe<math>\TeX</math></b>	171
13.1	Con $\TeX$ t	171
13.2	Lua $\TeX$	171
13.3	te $\TeX$	172
13.4	Xe $\TeX$	172
<b>14</b>	<b>Bib<math>\LaTeX</math> and Biber</b>	173
14.1	Bib $\TeX$ Problems	173
14.2	Solutions	173
14.3	Differences Between Bib $\TeX$ and Bib $\LaTeX$	174
14.4	Migrating from Bib $\TeX$ to Bib $\LaTeX$	174
14.4.1	$\LaTeX$ syntax	174
14.4.2	Bib Syntax	175
<b>15</b>	<b>Using <math>\LaTeX</math> Syntax as an Unambiguous Way to Communicate</b>	177

<b>16</b>	<b>Fonts' encodings</b>	179
16.1	Do not Mix the Different Encodings	179
16.2	Brief History	179
16.3	'Type' Fonts	179
16.4	Bitmap vs Outline	180
16.5	The OT1 Encoding	180
16.6	Hinting	181
16.7	The Cork Encoding: T1	181
16.7.1	Description	181
16.7.2	Together With a Font Package	181
16.8	Type 1, Type 3, TrueType and OpenType	182
16.9	PDF <sub>TEX</sub>	183
16.10	Practical Considerations	183
16.11	Converting .ps Files That Were Produced With CM Bitmap Fonts	184
16.12	Accents Encoding	184
16.13	Summary	185
16.14	Internationalization	185
16.15	Documents as a Source of Information	185
<b>17</b>	<b>A Student Point of View on the Aesthetics of Publications, from the Perspective of Effectiveness</b>	187
17.1	Introduction	187
17.2	What You Need to do to Write a Good Article	188
17.2.1	Use L <sup>A</sup> T <sub>E</sub> X	188
17.2.2	Structure	188
17.2.3	Write Concisely	189
17.2.4	Write Precisely	189
17.2.5	Write Adapted	189
17.2.6	Write Univocal	190
17.2.7	Correct, but Later	190
17.3	What L <sup>A</sup> T <sub>E</sub> X Brings You	190
17.3.1	Structure	190
17.3.2	Automatization	190
17.3.3	Dynamic Referencing	191
17.4	Graphical Aesthetism	191
17.4.1	Why Not	191
17.4.2	But Not Too Much	191
17.5	Writing Aesthetism	194
17.5.1	Emphazing Something	194
17.5.2	Choosing a Font	195
17.5.3	Using fancyhdr	195

<b>18 eBooks</b>	197
18.1 eBooks: Why?	197
18.2 L <sup>A</sup> T <sub>E</sub> X and eBooks	198
18.2.1 Why?	198
18.2.2 How?	198
18.2.3 Is PDF the Right Format?	203
18.3 Conclusion	203
<b>19 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> Learning Curve</b>	205
19.1 Most Common Beginners' Mistakes	206
<b>20 Some Useful L<sup>A</sup>T<sub>E</sub>X-Related Webservices</b>	209
<b>21 L<sup>A</sup>T<sub>E</sub>X Package Conflicts</b>	211
<b>22 Test Crossword</b>	215
<b>References</b>	217
<b>Index</b>	227

---

## List of Figures

1.1	Circuit with code 2. ....	33
1.2	Circuit with code 1. ....	33
1.3	Four mathematical symbols, the one at the right bottom corner being defined as the ‘product integral’ symbol. ....	79
1.4	A caption ....	86
3.1	Upper and lower case forms of the $\text{III}$ symbol. ....	121
3.2	The $\text{III}(t)$ ‘function.’ ....	121
6.1	How to put your C6 envelope in the printer. ....	138
6.2	C6 generated envelope. ....	139
12.1	A Mandelbrot fractal generated using $\text{\LaTeX}$ . ....	169

This page intentionally left blank.

---

**List of Tables**

1.1 Principal logic gates..... 39

3.1 Some math outputs..... 117

This page intentionally left blank.

---

## Preface

### First Edition

L<sup>A</sup>T<sub>E</sub>X is a beautiful programming language. One sometimes wants to write in an easy and quick way various documents, such as letters, papers, reports, theses, and so on. Here, “one” can be a new student, an old professor, or even somebody who is not linked at all with any academic stuff.

We often want special constructs in our documents. This is more difficult than basic typesetting, especially using L<sup>A</sup>T<sub>E</sub>X when you are not experienced.

On one hand, scientists do not always have the time to learn another programming language, especially if they are making a PhD, or doing something else which asks for a lot of time. Using direct and working programming tricks can speed up your productivity, and make you learning better. But where can they be found?

On the other hand, everybody wants to make nice documents, if possible. A nicer document attracts the reader. Evidently, “a *nice* document” is a subjective notion. By “nice document,” we want to speak about a document which

1. is structured in an unambiguous way, and has a clear presentation,
2. is read with pleasure, if possible.

The beginner, whoever he is, often faces strange errors because he tries to teach L<sup>A</sup>T<sub>E</sub>X to do what he wants it to do. It is difficult to learn the advanced notions about L<sup>A</sup>T<sub>E</sub>X, even in an average period of time. The good books have to be found. But that is often the moment where you lack of time and that you are happy to find complete solutions. That is a moment where you tell to yourself “Is there no folk out there to give me such a trivial tip?” but nobody hears it. The aim of this book, is to avoid you these problems: you do not need to write everything from scratch, ask on various forums or Usenet groups, or even read heaps of books just to make a simple script. That is my goal. You can now continue writing your document in L<sup>A</sup>T<sub>E</sub>X’s philosophy: concentrating on your document’s content, at the place of its appearance.

These reasons justify the presence of this book. I have tried to pack the most useful tricks which can enhance your documents' content in a concise way. The aim is not to teach L<sup>A</sup>T<sub>E</sub>X programming, but to give a quick reference to all the tips and tricks that can be used if you are encountering a (difficult) problem, or simply facing a question which you cannot find the answer to.

I consider that the reader is familiar with basic L<sup>A</sup>T<sub>E</sub>X programming. It can be learnt from various sources: [22, 28, 47, 55, 59, 97, 165, 175], and many others.

In many book(let)s, a part is always devoted to tell how the reader should read the book. It is, at least to me, often too directive. Here is the way I propose you to read this booklet. Firstly, have a look at the table of contents. Many different subjects are treated in this booklet, and reading them from one to another, until the end, could be really boring. You may, after having read the table of contents, roughly check the two parts, and then begin to read what you need to read. If you simply want to learn something about a particular subject, you will find this booklet very useful. On the other hand, if you want to read everything, you can, but the structure of the document has not been thought for such a reading scheme. Depending on your degree of knowledge with L<sup>A</sup>T<sub>E</sub>X, you can

1. either read everything, because you do not know a lot about this,
2. or read particular subjects, referring to the outline, to deepen your knowledge.

I hope this will give you new ideas about advanced ways to typeset text and to make different processes automatic, thus giving you more time for other activities.

If you have any suggestion, whatever the topic, feel free to send me an e-mail at

[mluca@swing.be](mailto:mluca@swing.be)

Liège,

*Luca Merciadri*  
October, 2009

## Second Edition

I am pleased to announce a second edition of this booklet. Most of the modifications come from my TUGboat articles. Here is a list of these modifications since the First Edition:

- ♡ Chapters 11 and 12 have been added;
- ♡ Section 1.10 has been added;
- ♡ Sections 2.9, 9.1 and 10.1 have been greatly improved;
- ♡ Subsections 1.2.2, 1.5.7, 1.5.9, 1.5.10, 1.5.12, 1.5.16, 1.7.4, 1.10.1, 1.14.6, 1.14.11, 1.14.16, 1.14.18, 1.14.23, 1.14.26, 1.14.28, 1.14.29, 2.3.1, 2.3.2, 2.4.1, 2.4.3, 2.5.1, 2.7.3 and 2.7.4 have been added;
- ♡ Subsubsections in Subsection 1.3.1 and in Subsection 1.14.21 have been added.

Liège,

*Luca Merciadri*  
Winter, 2010

## Third Edition

I am pleased to announce a third edition of this booklet. Most of the modifications come from my TUGboat articles. Here is a list of these modifications since the Second Edition:

- ♡ Chapters [13](#), [14](#), [15](#), [16](#), [17](#), [18](#) and [22](#) have been added;
- ♡ Sections [1.1](#), [3.4](#) and [16.15](#) have been added;
- ♡ Subsections [1.4.6](#), [1.5.11](#), [1.5.15](#), [1.5.17](#), [1.11.2](#), [1.12.1](#), [1.14.1](#), [1.14.2](#), [1.14.5](#), [1.14.8](#), [1.14.9](#), [1.14.13](#), [1.14.17](#), [1.14.20](#), [1.14.21](#), [1.14.24](#), [1.14.31](#), [1.14.35](#), [1.14.39](#) and [2.7.7](#) have been added.

Liège,

*Luca Merciadri*  
Spring, 2011

## Fourth Edition

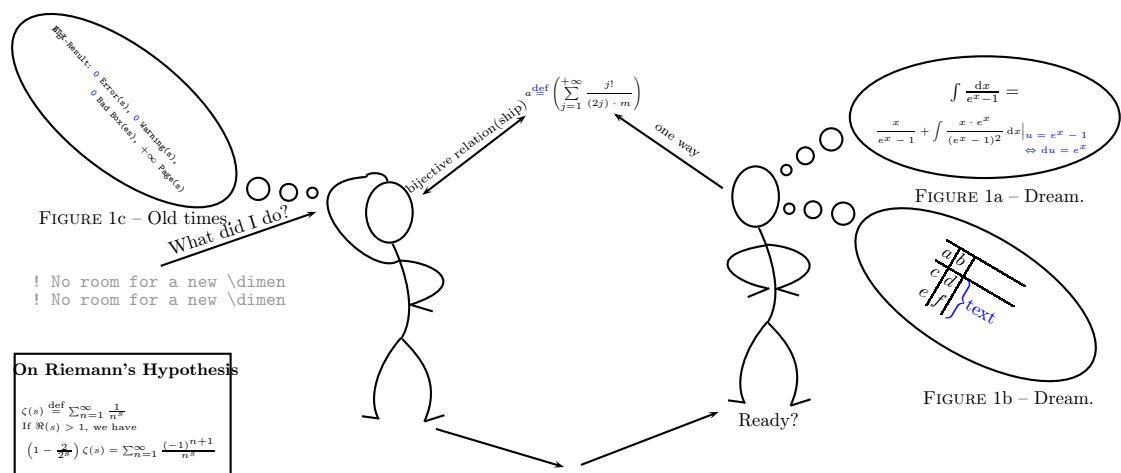
I am pleased to announce a fourth edition of this booklet. Most of the modifications come from my TUGboat articles since third edition. Here is a list of these modifications since the Third Edition:

- ♡ Chapters 19, 20 and 21 have been added,
- ♡ Sections 3.1, 3.2, 3.3, 3.5, 3.6, 3.7, 3.8, 3.9 and 3.10 have been added,
- ♡ Subsections 1.4.2, 1.4.3, 1.5.1, 1.5.2, 1.5.3, 1.5.4, 1.5.5, 1.5.6, 1.5.8, 1.13.1, 1.13.2, 1.14.3, 1.14.32 and 9.1.1 have been added.

Liège,

*Luca Merciadri*  
Winter, 2011

This page intentionally left blank.



On Riemann's Hypothesis

$\zeta(s) \stackrel{\text{def}}{=} \sum_{n=1}^{\infty} \frac{1}{n^s}$   
If  $\Re(s) > 1$ , we have  
 $(1 - \frac{2}{2^s}) \zeta(s) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n^s}$

...

■

FIGURE 2 – Your final paper (commonly called “Reality”).

```
\shortstack[c]{u=e^x-1}\Leftrightarrow \mathrm du=e^x\text{ and }x=\ln(u+1)

\begin{tabular}{l|l|r}
\hline $a$ & $b$ & \\
\hline $c$ & $d$ & \multirow{2}{*}{\rdelim\}{2}{1cm}[text]\$e$ & $f$ & \\
\hline
\end{tabular}

\stackrel{\text{tiny def}}{=}

\makeatletter
\def\Biggg#1{\hbox{$\left#1\mathrm to32\mathrm p\right#1$}\right.\n space}}
\newdimen\bracketwidth \settowidth{\bracketwidth}{\Biggg{}}
\makeatother

\relax!
```

\relax!

This page intentionally left blank.

## Standard Documents



We call a *standard document* a document like a paper, a report, a thesis, any other academic work, or even a document which is not academic.

Let's say you are writing such documents for some reasons, and that you progressively want to modify their structure so they become nicer. You now have a practical experience with L<sup>A</sup>T<sub>E</sub>X but there are some kinds of things which do not work as you want them to, and they are important:

1. Margins are too large, and you want to modify them so you can put more text on a page,
2. You encounter errors with `\dimen`, but you do not know what it is,
3. Your captions are too big. You have to modify them,
4. You need to specify that your work has been funded by an organization, but you do not want a footnote number to appear,
5. You have to give examples of logic gates, but you never found a good way to do this easily,
6. You think that it would be better to set minitocs where you want, so your reader will be more familiar with your Chapters / Sections / Subsections, ..., but you do not know how to make this,
7. As your document will be posted on the Internet, it would be interesting to make URLs becoming links. Furthermore, you think that allowing clicking on various links in your document will improve the reader's experience with your document,
8. Your document is in English, but your abstract would be nice in French too,
9. Other persons in your department are fierce of their L<sup>A</sup>T<sub>E</sub>X titlepages but you are too shy (or simply do not want) to ask them how to make this,
10. You use many strange math operators which are not even built-in, and you want them to be implemented,
11. You would want to know how to make different enumerations,
12. Counting rows in tables would be interesting for you,
13. Putting some text under an image would be nice,
14. You think boxed minipages are really beautiful, but you believe it is really tricky to make this,
15. You wonder how to put many expressions in a subscript, like a kind of stack,
16. You would like to underline and dash text, and you wonder why it is not already implemented in L<sup>A</sup>T<sub>E</sub>X,
17. You would like to do strange (and pedagogical) things with matrices: displaying text over them, for example, or putting braces,
18. You are fed up with traditional L<sup>A</sup>T<sub>E</sub>X slides and want to try other solutions,
19. You would ask for putting braces in tables,
20. You have done a lot of MATLAB graphics, but you would be disappointed if they would not be able to be put in your L<sup>A</sup>T<sub>E</sub>X documents.

Even if you are able to do these things,

1. You need to typeset units from the SI, but you think there is a better way to typeset these units, than doing `\mathrm{myunit}` and playing with spaces,
2. You would know how many pages and tables there are in your document, but you do not know how to automatically count this,
3. You think that the traditional QED symbol is ugly, or you are fed up with it, as you read so many proofs that you simply want a “*different way*” to show things,
4. You would like to know how to manage indentation,
5. You need to typeset strange symbols you do not even know how they are pronounced (that is, are you working Applied Sciences or in Mathematics?),
6. As in every book you read, you think that displaying a message on would-be blank pages would be nice,
7. You would like to avoid the (here) unserious blue color in your TOC and TOFs,
8. You would like to learn some tricks to make your maths seeming more serious than these of your friends,
9. You have to put crops on your pages, but you do not know how,
10. You never wrote a big document in L<sup>A</sup>T<sub>E</sub>X, and you wonder how to manage this,
11. You would like your document to look better than the other ones of your friends,
12. You would like to learn some pedagogical tools that you can implement with L<sup>A</sup>T<sub>E</sub>X,

but

1. You do not know how to do these things,
2. You think that some of the things you want to do are so weird than nobody ever thought about the way to implement them,
3. You do not have a lot of time, and you want these things to be realized as soon as possible,
4. You can do them alone, but you need a support.

If so, you are ready to read this book. There are other examples, but everything is not going to be unveiled here!

We shall classify these problems in two categories. The first one will be devoted to the first items, considered as *major tricks*. The second will be devoted to *fine tuning* of things.

We shall take a deep look at both categories. We begin by the major tricks which can be applied to your simple documents.

## Major Tricks

There are sometimes some tricks that you should know, because they are unavoidable. These are the ones which make the difference between a guy who learnt the basics of  $\text{\LaTeX}$  in two or three days and *you*. They will allow you to make more “intelligent” documents: using counters, showing (more pedagogical) mathematical expressions without any language’s limitation, and more generally, using the advantages of  $\text{\LaTeX}$  without being limited by its complexity.

### Contents

---

<b>1.1</b>	<b>Allowing</b>	<b>10</b>
1.1.1	Linebreaks After Comma in Math Mode	10
<b>1.2</b>	<b>Avoiding</b>	<b>11</b>
1.2.1	Erroneous Logic Formulae	11
1.2.2	Erroneous References for Floats	12
<b>1.3</b>	<b>Counting</b>	<b>14</b>
1.3.1	Introduction	14
	Using Counters: Why?	14
	Creating a Counter	14
	Modifying a Counter’s Value	15
	Printing a Counter’s Value	15
1.3.2	Equations For an Appendix	16
1.3.3	Examples	16
	Counting Sections in Roman	16
1.3.4	Rows In Tables	16
<b>1.4</b>	<b>Creating</b>	<b>17</b>
1.4.1	Counters	17
1.4.2	Enumerate Lists With a Star	17
	Code	17
	Examples	18
1.4.3	Math Math Operators	18
	Code	18
	Example	19
1.4.4	Math Operators	19

1.4.5	New Abstract Environments	20
	Example	21
1.4.6	Quotation Marks Using ”	21
1.4.7	Titlepages	22
	Example	24
	Finding Other Titlepages	24
<b>1.5</b>	<b>Drawing</b>	<b>24</b>
1.5.1	A Block Quote With Quotation Marks	24
	Code	25
	Example	25
1.5.2	A Box On Every Digit Of an Integer	25
	Code	26
	Example	26
1.5.3	A Centered Slash on Math Symbols	26
1.5.4	A Colored Front Cover	27
	Code	27
	Example	27
1.5.5	A (Single/Double) Linked List	28
1.5.6	A Personalized <code>overline</code>	29
	Code	29
	Example	30
1.5.7	Boxes In <code>align</code> -like Environments	30
1.5.8	Cases With Square Brackets	31
	Code	31
	Example	32
1.5.9	Closed Square Roots	32
1.5.10	Correct Circuits	32
1.5.11	Dependency Arrows	34
	Code	34
	Result	34
1.5.12	Diagonally-Cut Cells	34
1.5.13	Graph Paper	34
	Simplest Alternative	35
	Trickiest Alternative	36
1.5.14	Logic Gates	37
	Independent Drawing	37
	Example	39
	Linking Them	39
1.5.15	Over and Under Braces on Same Elements	42
1.5.16	Rightmost Braces	43
1.5.17	Tables With Dashed Lines, Shaded Cells and Arrows Pointing At a Row and a Column	44
<b>1.6</b>	<b>Finding</b>	<b>45</b>
1.6.1	Symbols’ Commands	45
<b>1.7</b>	<b>Fixing Errors</b>	<b>45</b>
1.7.1	Dimen	45
1.7.2	Fncychap	46
1.7.3	L <sup>A</sup> T <sub>E</sub> X	46
	fix-cm	46

fixltx2e .....	46
1.7.4 PDF compatibility .....	47
<b>1.8 Incorporating .....</b>	<b>47</b>
1.8.1 MATLAB Graphics .....	47
LaPrint .....	47
Matfig2PGF .....	48
Matlab2Tikz .....	49
1.8.2 Spreadsheets .....	50
1.8.3 References .....	50
<b>1.9 Modifying .....</b>	<b>50</b>
1.9.1 Captions .....	50
1.9.2 Margins and Marginpar Notes .....	50
1.9.3 The Zoom of a Document .....	52
<b>1.10 Printing .....</b>	<b>52</b>
1.10.1 Empty Pages .....	52
1.10.2 Monochrome .....	52
<b>1.11 Sizing .....</b>	<b>53</b>
1.11.1 Boxed Minipages Depending On the Text .....	53
1.11.2 \left and \right to Produce Smaller Brackets .....	53
<b>1.12 Splitting .....</b>	<b>54</b>
1.12.1 algorithm Environments on Multiple Pages .....	54
<b>1.13 Using .....</b>	<b>55</b>
1.13.1 \centering and the center Environment Accordingly .....	55
1.13.2 \scalebox .....	55
1.13.3 The Enumerate Package .....	56
<b>1.14 Writing .....</b>	<b>56</b>
1.14.1 Aligned Systems of (In)Equations .....	56
1.14.2 (a) Bold \ell .....	57
1.14.3 A Symmetric Matrix .....	57
1.14.4 Code Depending On the Processor .....	58
1.14.5 Dancing Text .....	59
1.14.6 Dash Integrals .....	60
1.14.7 Dashed And Underlined Text .....	60
1.14.8 Date and Time .....	61
1.14.9 Diagonal (Inverse) Dots .....	61
1.14.10 Different Slides .....	62
1.14.11 Enumerations With Textcircled Numbers .....	63
1.14.12 Footnotes In Boxes .....	63
Example .....	63
Code .....	64
1.14.13 Footnotes In tabular .....	65
1.14.14 Footnotes With a New Number At Each Section .....	65
1.14.15 Footnotes Without Any Number .....	66
1.14.16 Fractions .....	66
1.14.17 Indented Text, But Not the First Line .....	66
1.14.18 Integrals Adapted to the Context .....	67
Matrices With Five Rows .....	67
Matrices With Four Rows .....	68

Matrices With Three Rows .....	68
Matrices With Two Rows .....	69
Matrices With One Row .....	69
The <b>Bigints</b> Package .....	70
1.14.19 Links .....	70
Avoiding URLs In the Margin .....	71
1.14.20 Logic Inferences .....	71
1.14.21 Matrices .....	72
Surrounded by... ..	72
With Braces .....	73
With Borders .....	75
With Full Borders .....	76
With Dots .....	76
With Text .....	77
With Same Space Between Elements When \boxed is Used .....	77
1.14.22 Minitocs .....	78
1.14.23 Product Integrals .....	79
1.14.24 Roman Numbers .....	79
1.14.25 Simplifications .....	80
1.14.26 Small <b>mathcal</b> Letters .....	80
1.14.27 Stacks (In Equations) .....	81
First Kind .....	81
Second Kind .....	81
1.14.28 Standard Display of Date and Time .....	82
1.14.29 Subequations with Braces .....	82
1.14.30 Tables .....	83
With Braces .....	83
1.14.31 Tabular Packages: What are Their Respective Purposes, and Which Ones Conflict? .....	83
Respective Purposes .....	84
Packages Conflicts .....	85
1.14.32 Text On the Same Line as an Equation .....	85
1.14.33 Text Below an Image .....	86
1.14.34 Two Itemizations on the Same Line .....	86
1.14.35 Verbatim in Center .....	87
1.14.36 Verbatim In <b>tabular</b> .....	87
1.14.37 Verbatim In Footnotes .....	87
1.14.38 Verbatim In Hierarchy .....	87
1.14.39 Vertically Aligned Itemize Environments .....	88

## 1.1 Allowing

### 1.1.1 Linebreaks After Comma in Math Mode

When L<sup>A</sup>T<sub>E</sub>X is in math mode, linebreaks are not allowed. It might be a problem when you're typesetting maths. However, it is possible to avoid this:  $\frac{x+1}{2}, 5x^2 + 3x - 4, \sqrt{6x-9}, \frac{x-7}{8}, \frac{x+1}{2}, 5x^2 + 3x - 4, \sqrt{6x-9}, \frac{x-7}{8}, \frac{x+1}{2}, 5x^2 + 3x - 4, \sqrt{6x-9}, \frac{x-7}{8}, \frac{x+1}{2}, 5x^2 + 3x - 4, \sqrt{6x-9}, \frac{x-7}{8}, \frac{x+1}{2}, 5x^2 + 3x - 4, \sqrt{6x-9}, \frac{x-7}{8}$ . This is what you get using [11]

```

\makeatletter
\def\old@comma{,}
\catcode'\,=13
\def,{%
  \ifmmode%
    \old@comma\discretionary{}{}{}%
  %   \old@comma$ $%
  %   $\old@comma\ $%
  \else%
    \old@comma%
  \fi%
}
\makeatother

```

If this code had not been used,

```

$\frac{x+1}{2}, 5x^2+3x-4, \sqrt{6x-9}, \frac{x-7}{8},
\frac{x+1}{2}, 5x^2+3x-4, \sqrt{6x-9}, \frac{x-7}{8},
\frac{x+1}{2}, 5x^2+3x-4, \sqrt{6x-9}, \frac{x-7}{8},
\frac{x+1}{2}, 5x^2+3x-4, \sqrt{6x-9}, \frac{x-7}{8},
\frac{x+1}{2}, 5x^2+3x-4, \sqrt{6x-9}, \frac{x-7}{8}$.

```

would have resulted in an ugly output: math formulae would have been printed even after the (column) margin.

## 1.2 Avoiding

### 1.2.1 Erroneous Logic Formulae

Let us express the dual of  $A$  as  $\overline{A}$ . The major problem using `\overline` is that if you write things like

$$\overline{AB}$$

(`\overline{A}\overline{B}`), they look exactly the same as

$$\overline{AB}$$

(`\overline{AB}`).

Or the latter is equal to  $\overline{A} + \overline{B}$  by de Morgan's rules, and thus differs from the former on a truth table. So, a distinction needs to be made. Enrico Gregorio gave me the solution. Implement

```

\newcommand{\closure}[2][3]{{}\mkern#1mu\overline{\mkern-#1mu#2}}

```

and then write

```

\closure{A}\closure{B}

```

and compare it to

```
\closure{AB}
```

The respective results are

$$\overline{AB} \quad \text{and} \quad \overline{AB}.$$

One can now see the difference. The optional argument to `\closure`, such as `\closure[2]{A}`, is meant to correct possible ‘errors’ due to letter form. [29]

### 1.2.2 Erroneous References for Floats

When writing a paper with  $\text{\LaTeX}$ , the authors often let  $\text{\LaTeX}$  do the cross-reference work. This results in a notable gain of time, because the work for every reference is automated. Consider a reference  $r$  declared using `\label{r}`. If  $r$  is cited,  $\text{\LaTeX}$  will

1. Know its page number, which can be displayed and linked (if `hyperref` is used) using `\pageref{r}`,
2. Know its reference, meaning that it knows  $r$ ’s place in the document structure.

But consider now a `tabular` environment placed in a `table` environment. Placing the `tabular` environment centered at the page is a good idea, thereby using

```
\begin{table}
  \begin{center}
    \begin{tabular}{cc}
      Text & Text
    \end{tabular}
  \end{center}
\end{table}
```

or its `\centering` variant. To link this table to a reference, one needs to place a `\label{reference}` in the `table` environment. One thing to remember is that `\label{}` always comes after `\caption{}`. That is, you cannot use neither

```
\begin{table}
  \label{reference}
  \caption{Name of the table.}
  \begin{center}
    \begin{tabular}{cc}
      Text & Text
    \end{tabular}
  \end{center}
\end{table}
```

nor

```

\begin{table}
  \begin{center}
    \begin{tabular}{cc}
      Text & Text
    \end{tabular}
  \end{center}
\label{reference}
\caption{Name of the table.}
\end{table}

```

You also need to end the `center` environment before using `\caption{}`. That is, you should not use

```

\begin{table}[!h]
  \begin{center}
    \begin{tabular}{cc}
      Text & Text
    \end{tabular}
  \caption{Name of the table.}
  \end{center}
\label{tab:test}
\end{table}

```

but rather use

```

\begin{table}[!h]
  \begin{center}
    \begin{tabular}{cc}
      Text & Text
    \end{tabular}
  \end{center}
\caption{Name of the table.}
\label{tab:test}
\end{table}

```

Notice also the better reference: `tab:test` is clearer than `reference`. As `\centering` is local to the (most nested) environment which contains it, you can evidently replace the `center` environment by a simple `\centering`:

```

\begin{table}[!h]
  \centering
  \begin{tabular}{cc}
    Text & Text
  \end{tabular}
\caption{Name of the table.}
\label{tab:test}
\end{table}

```

This concept is very important: some classes will not render a reference if the `\caption{}–\label{}` order is not respected. Even worse, others will put unrelated reference numbers, such as `\thesection`, which can be disastrous: writing “thanks to Theorem  $x$ , we have [...]” is a good way not to lose the reader, but if  $x$  is a theorem number which does not exist, or which has no link with the citation, the whole paper might seem hastily written, or at least not edited, or simply confusing to the reader.

## 1.3 Counting

*Counting* is, in a general way, useful for various purposes. It is difficult to avoid using it, for example when you need to buy something. That is the same with  $\text{\LaTeX}$ : the whole  $\text{\LaTeX}$  programming language is based on counters.

### 1.3.1 Introduction

#### Using Counters: Why?

$\text{\LaTeX}$  uses a lot of counters. For example, here are some counters:

- |             |               |                   |
|-------------|---------------|-------------------|
| 1. chapter  | 7. figure     | 13. section       |
| 2. enumi    | 8. footnote   | 14. subparagraph  |
| 3. enumii   | 9. mpfootnote | 15. subsection    |
| 4. enumiii  | 10. page      | 16. subsubsection |
| 5. enumiv   | 11. paragraph | 17. table         |
| 6. equation | 12. part      |                   |

They are used for numbering elements in the `enumerate` lists, equations, figures, sections and related, tables, ...

For example, when you use the `enumerate` environment, the hierarchy you would like to appear on your page is directly linked to `enumi` (and related) counters. Without them, it would be impossible to number your elements in an easy way, as it would be impossible for you to pay securely at a traditional shop. Lots of operations can also be applied to them. These operations constitute the main concern of the following subsections.

#### Creating a Counter

If you want to *define a new counter* `mynewcounter`, just use

```
\newcounter{mynewcounter}
```

Note that you can ask  $\text{\LaTeX}$  to reset `mynewcounter`’s value whenever the value of a counter `dummycounter`, is incremented:

```
\newcounter{mynewcounter}[dummycounter]
```

### Modifying a Counter's Value

If you want to add a value  $x$  (negative or not) to a counter `mycounter`, just use

```
\addtocounter{mycounter}{x}
```

If you want to set a counter `mycounter` to a value  $x$  (negative or not), just use

```
\setcounter{myvalue}{x}
```

There are also cases where a counter is already (pre-)defined, but it is not incremented as you want. [98] For example, things such as figures, tables and footnotes are automatically numbered “by chapter” when you use the standard *book* and *report* classes. The process of resetting is done automatically, when the “master” counter is stepped (when the `\chapter` command that starts chapter  $n$  happens, the chapter counter is stepped, and all the dependent counters are set to zero).

It might be interesting to do this for algorithms, for example. To define this by hand, you declare the relationship directly at the counter's definition:

```
\newcounter{mynewcounter}[mastercounter]
```

Using this, every time counter `mastercounter` will be stepped, `mynewcounter` will be reset. This command is equivalent to

```
\makeatletter
\@addtoreset{mynewcounter}{mastercounter}
\makeatother
```

In these cases, `mynewcounter` is said to be a *slave counter*, the `mastercounter` being the *master counter*.

### Printing a Counter's Value

If you want to *print* the **arabic value** of a built-in counter, just use `\thecounter`, where `counter` is the name of the counter. If it is needed in a more complicated situation, such as some **internal arithmetic**, use

```
\value{thecounter}
```

You can also print the `thecounter`'s value in arabic using

```
\arabic{thecounter}
```

If `thecounter`'s value is less than 27, you can print its **alphabetical equivalent** using

```
\alph{thecounter}
```

to use *lower case* alphabetic letters, and

```
\Alph{thecounter}
```

to use *upper case* alphabetic letters.

You can print it in **roman numerals**, using

```
\roman{thecounter}
```

For example, you can use the `enumerate` package together with

```
\begin{enumerate}[\textcircled{\arabic{enumi}}]
\item myitem1
\item myitem2
\end{enumerate}
```

to have

① myitem1
② myitem2

### 1.3.2 Equations For an Appendix

If you want all your equations' numbers in your appendix to be printed with a **A-** before them, just use

```
\renewcommand{\theequation}{A-\arabic{equation}}
```

### 1.3.3 Examples

#### Counting Sections in Roman

If you want your `\section{}`'s output to be given in Roman, just use

```
\renewcommand{\thesection}{\Roman{section}. Section \Roman{section}}
```

in the preamble. Then, you can use `\section{}` with interest. For example, if `section` equals 1, your first section will be displayed "I. Section I." It can be easily adapted for another element of the hierarchy.

### 1.3.4 Rows In Tables

A traditional application of tables is to number every row in it. If you put

```
\usepackage{array}
```

in the preamble, and

```
\newcounter{rowno}
\setcounter{rowno}{0}
\begin{document}
\begin{tabular}{>{\stepcounter{rowno}\therowno.}c1}
\multicolumn{1}{r}{No.} & text \\\
```

```

\hline
& first  \\
& second \\
& third  \\
& fourth
\end{tabular}

```

in the body, the result is the following:

No. text
1. first
2. second
3. third
4. fourth

## 1.4 Creating

### 1.4.1 Counters

See [1.3](#), from p. [14](#).

### 1.4.2 Enumerate Lists With a Star

You might want to add an asterisk after some given labels in the `enumerate` environment. I here give some solutions from [\[130\]](#).

#### Code

For single-level enumerations, you might simply use Martigan's method:

```

\begin{enumerate}
  \item[1. ] First
  \item[2.*] Second
\end{enumerate}

```

Notice the extra space after the 1. in order to align 1 and 2 vertically.

For more complicated structures, consider defining a new (modified `enumerate`) environment as `egreg` did:

```

\newenvironment{modenumerate}
  {\enumerate\setupmodenumerate}
  {\endenumerate}

\newif\ifmoditem
\newcommand{\setupmodenumerate}{%
  \global\moditemfalse

```

```

\let\origmakelabel\makelabel
\def\moditem##1{\global%
  \moditemtrue\def\mesymbol{##1}\item}%
\def\makelabel##1{%
  \origmakelabel{##1\ifmoditem%
    \rlap{\mesymbol}\fi\enspace}%
  \global\moditemfalse}%
}

```

You can now use it:

```

\begin{modenumerate}
\item First
\moditem{*} Second
\end{modenumerate}

```

## Examples

You can notice that both solutions give the same result.

Using the first method, one gets

1. First
- 2.\* Second

The second method gives

1. First
- 2.\* Second

Note that it is also possible to put asterisks before the labels, as e.g. detailed at [\[130\]](#) too.

### 1.4.3 Math Math Operators

There is no typo. Here is how to type a ‘Math Math Operator,’ i.e. some operator both in the  $\text{\LaTeX}$  and in the Math. senses.

Several ways to create your big own math operator (say a big ‘A’ in this example) were proposed at [\[141\]](#).

## Code

Let us consider two of all the proposed solutions:

```

\newcommand{\opA}{\mathop{\vphantom{\sum}}}%
\mathchoice
{\vcenter{\hbox{\huge A}}}%
{\vcenter{\hbox{\Large A}}}{\mathrm{A}}%
{\mathrm{A}}\displaylimits}

```

(thanks to `egreg`) and

```
\def\Aop{\operatornamewithlimits{%
  \mathchoice{\vcenter{\hbox{\huge A}}}%
               {\vcenter{\hbox{\Large A}}}%
               {\mathrm{A}}%
               {\mathrm{A}}}}
```

(thanks to Herbert).

Note that

```
\DeclareMathOperator*{\Aop}{A}
```

allows you to define a  $A$  operator, but which will not be scaled to some big size:  $\overset{n}{A}$ .

### Example

Successively calling `\Aop` and `\opA` gives, in an `equation` environment:

$$\overset{n}{A} \quad \text{and} \quad \overset{n}{A}$$

Inline math gives, using the same successive calls:  $A_{i=1}^n$  and  $A_{i=1}^n$ .

#### 1.4.4 Math Operators

*Math operators* are basically functions in a more global sense than the mathematical one: they take a list of argument (which is not void), and return something.

There is a lot of ways to declare new math operators. The best way to do it is to use

```
\DeclareMathOperator{\theoperator}{TheOperator}
```

For example, here are some useful math operators:

```
\DeclareMathOperator{\adj}{adj}
\DeclareMathOperator{\arccosh}{arccosh}
\DeclareMathOperator{\arccotan}{arccotan}
\DeclareMathOperator{\arcsinh}{arcsinh}
\DeclareMathOperator{\cis}{cis}
\DeclareMathOperator{\cod}{c}
\DeclareMathOperator{\com}{com}
\DeclareMathOperator{\cosec}{cosec}
\DeclareMathOperator{\cosech}{cosech}
\DeclareMathOperator{\cotan}{cotan}
\DeclareMathOperator{\cotanh}{cotanh}
```

```

\DeclareMathOperator{\decod}{d}
\DeclareMathOperator{\diag}{diag}
\DeclareMathOperator{\dist}{dist}
\DeclareMathOperator{\Div}{div}
\DeclareMathOperator{\dom}{dom}
\DeclareMathOperator{\domc}{dom_{c}}
\DeclareMathOperator{\domd}{dom_{d}}
\DeclareMathOperator{\Ei}{Ei}
\DeclareMathOperator{\erf}{erf}
\DeclareMathOperator{\Gal}{Gal}
\DeclareMathOperator{\grad}{grad}
\DeclareMathOperator{\id}{id}
\DeclareMathOperator{\imf}{im}
\DeclareMathOperator{\Ind}{Ind}
\DeclareMathOperator{\li}{li}
\DeclareMathOperator{\Li}{Li}
\DeclareMathOperator{\pgcd}{pgcd} % for the french equivalent of ‘‘GCD’’
\DeclareMathOperator{\res}{res}
\DeclareMathOperator{\rot}{rot}
\DeclareMathOperator{\Sa}{Sa}
\DeclareMathOperator{\sech}{sech}
\DeclareMathOperator{\sech}{sech}
\DeclareMathOperator{\sign}{sign}
\DeclareMathOperator{\sinc}{sinc}
\DeclareMathOperator{\trace}{tr}

```

You can now use, for example,  $\text{dom}(f(x))$ , just by writing

```
\dom(f(x))
```

#### 1.4.5 New Abstract Environments

Let’s say you want to use a french abstract environment, but not necessarily at the place of the traditional `abstract` environment. That is, you want two different abstracts *environnements*. It is easily achieved using

```

\usepackage[frenchb,english]{babel}
\newcommand*{\abstractname}{Résumé}
\newenvironment{abstract}{%
  \small
  \begin{center}
    {\bfseries \abstractname\vspace{-.5em}\vspace{0pt}}
  \end{center}
  \quotation
}
{\endquotation}

```

in the preamble. We have chosen `frenchb` because, according to [20], `babel` being installed on the L<sup>A</sup>T<sub>E</sub>X<sub>2</sub><sub>ε</sub> distributions, you are sure that the result does not depend on your configuration. For example, using

```
\usepackage[french,english]{babel}
```

at the place of

```
\usepackage[frenchb,english]{babel}
```

and thus replacing `frenchb` by `french`, does not have the same consequences on all the different configurations. Note that the last chosen language is kept in `\bbl@main@language`, and is thus the main language of the document.

It is now easy to change between two abstracts. Let's switch between French and English:

```
\selectlanguage{français}
\begin{abstract}
Votre r\'esum\'e.
\end{abstract}
\selectlanguage{english}
\begin{abstract}
Your abstract.
\end{abstract}
```

which needs to be put in the preamble.

### Example

The given code results in this text:

<b>Résumé</b> Votre résumé.
--------------------------------

#### 1.4.6 Quotation Marks Using ’’

Despite this might not be a good way to achieve correctly opening and closing quotation marks, that e.g. need to be typeset using ‘‘ (opening) and ’’ (closing) for English, you can manage to make the ’’ key achieve this. Simply use [11]

```
\makeatletter
\catcode'\''=13
\let\@hyphen@open\@empty
\def"{%
  \ifx\@hyphen@open\@empty%
    \def\@hyphen@open{a}%
    ‘‘%
  \else
```

```

    ’,%
    \let\@hyphen@open\@empty%
    \fi%
  }
  \makeatother

```

However, it needs to be understood that doing this will cause unappropriated characters when using " for e.g. umlauts.

#### 1.4.7 Titlepages

The *titlepage* is the first page that will be seen on your document. It has an influence on the (potentially future) reader. Doing it with care is a good but difficult thing. However, one can define a standard titlepage such as

```

\begin{titlepage}
\begin{center}

% Upper part of the page
\textsc{\LARGE YOUR UNIVERSITY}\\[1.5cm]

\includegraphics[width=0.50\textwidth]{_img/your_university_logo.eps}\\[1cm]

% Title
%\HRule \\[0.4cm]
\rule{\textwidth}{1pt}\par
\vspace{0.50cm}
{\huge \bfseries The name of your book\ \Large -- in \ref{TotPages} pages,
with \AbsTables ~tables  --}\[0.4cm]
\rule{\textwidth}{1pt}\par
%\HRule \\[1.5cm]

\vfill

% Author
\Large{\textsc{FirstName\footnote{\href{mailto:FirstName.LastName@
provider.domain}{FirstName.LastName@provider.domain}} LastName}
\vfill

City, Country

\vfill

% Bottom of the page
{\large \today}

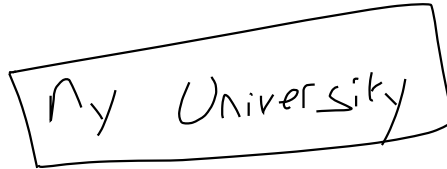
```

```
\end{center}  
\end{titlepage}
```

to put in your preamble, assuming the `hyperref`, `totpages` and `graphicx` packages have been loaded before. If `\AbsTables` outputs 1, you can still use the `ifthen` package to modify “tables” in “table,” automatically. The following example is voluntarily not in a box.

**Example**

YOUR UNIVERSITY



---

**The name of your book**  
– in 250 pages, with 2 tables –

---

FIRSTNAME<sup>1</sup> LastName

City, Country

October 7, 2011

**Finding Other Titlepages**

Peter Wilson has developped a collection of titlepages which can be found at [\[189\]](#).

**1.5 Drawing**

**1.5.1 A Block Quote With Quotation Marks**

Block quotes surrounded by quotation marks generally look beautiful, at least to me. Several solutions were proposed at [\[134\]](#).

---

<sup>1</sup> [FirstName.LastName@provider.domain](mailto:FirstName.LastName@provider.domain)

## Code

To take Mr. Munn's solution, its L<sup>A</sup>T<sub>E</sub>X code is

```
\usepackage{libertine}
\newcommand*\quotefont{\fontfamily{fxl}}

\usepackage{tikz}
\usepackage{framed}
% Make commands for the quotes
\newcommand*\openquote{\tikz[%
remember picture,overlay,%
xshift=-15pt,yshift=-10pt]
\node (OQ) {\quotefont%
\fontsize{60}{60}%
\selectfont‘‘};\kern0pt}
\newcommand*\closequote{\tikz[%
remember picture,overlay,%
xshift=15pt,yshift=10pt]
\node (CQ) {\quotefont%
\fontsize{60}{60}\selectfont’’};}
\definecolor{shadecolor}{%
named}{Azure}
\newenvironment{shadequote}%
{\begin{snugshade}%
\begin{quote}\openquote}
{\hfill\closequote\end{quote}\end{snugshade}}
```

You can then use the `shadequote` environment.

## Example

Using

```
\begin{shadequote}
A common mistake that people make%
when trying to design something%
completely foolproof is to%
underestimate the ingenuity%
of complete fools.%
\par\emph{Douglas Adams}
\end{shadequote}
```

gives (the rectangle has a light blue background)

### 1.5.2 A Box On Every Digit Of an Integer

It might be useful to box every digit of a given number.

“ A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools. ”  
*Douglas Adams*

## Code

This is achieved thanks to the following code, coming from Mr. Scharrer at [\[139\]](#).

```
\newcommand*\boxednumber}[1]{%
  \expandafter\readdigit\the\numexpr#1\relax%
  \relax
}
\newcommand*\readdigit}[1]{%
  \ifx\relax#1\else
    \boxeddigit{#1}%
    \expandafter\readdigit
  \fi
}
% Format macro used for every digit, adjust
% to your liking:
\newcommand*\boxeddigit}[1]{%
  \fbox{#1}\hspace{-\fboxrule}}
```

## Example

You can now use `\boxednumber{2011}` to get

$\boxed{2011}$ .

Other solutions for slightly different outputs are proposed at [\[139\]](#).

### 1.5.3 A Centered Slash on Math Symbols

Consider

$\nrightarrow$  and  $\nleftarrow$ .

The first is obtained using `\not`, when the second comes from the use of `\centernot`, from the `centernot` package. This avoids redefining a more or less centered `\not`. Thanks to `diabonas` for this answer at [\[146\]](#).

### 1.5.4 A Colored Front Cover

Lately, I wanted to create a front cover observing the following properties:

- colored,
- plain,
- aesthetically interesting.

I ended up using the `color` package and the `\pagecolor` command for the easiness of picking a color, and using it.

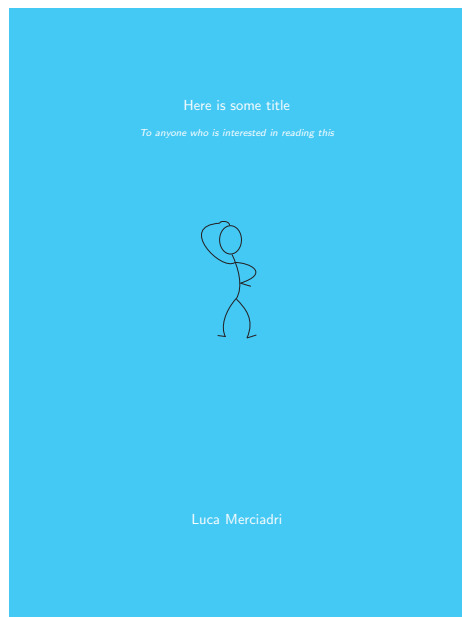
#### Code

The code is quite simple. Text is separated using fixed values.

```
\thispagestyle{empty}
\pagecolor[HTML]{33CCFF}
\begin{center}
\textcolor{white}{\sffamily %
  Here is some title}
\\[0.2cm]
\textcolor{white}{\sffamily %
  \scriptsize \itshape%
  To anyone who is interested in reading this}
\vspace{-7cm}
\begin{minipage}{\textwidth}
\scalebox{0.6}{
% image
}
\\
\centering
\textcolor{white}{\sffamily Luca Merciadri}
\end{minipage}
\end{center}
```

#### Example

It results in the following (scaled) front cover:



### 1.5.5 A (Single/Double) Linked List

Linked lists are a common tool in programming. Using a chain to place the nodes, and multipart rectangles for the double nodes, Mr. Gonzalo Medina managed at [138] to draw such an object using *Tikz*.

First load the *tikz* package, then

```
\usetikzlibrary{calc,shapes.multipart,%
chains,arrows}
```

The code to draw the thing is then

```
\begin{tikzpicture}[list/.style={%
rectangle split, rectangle split parts=2,
draw, rectangle split horizontal},%
>=stealth, start chain]

\node[list,on chain] (A) {12};
\node[list,on chain] (B) {99};
\node[list,on chain] (C) {37};
\node[on chain,draw,inner sep=6pt] (D) {};
\draw (D.north east) -- (D.south west);
\draw (D.north west) -- (D.south east);
\draw[*->] let \p1 = (A.two),
  \p2 = (A.center) in (\x1,\y2) -- (B);
\draw[*->] let \p1 = (B.two),
```

```

\p2 = (B.center) in (\x1,\y2) -- (C);
\draw[*->] let \p1 = (C.two),
\p2 = (C.center) in (\x1,\y2) -- (D);
\end{tikzpicture}

```

Here is the output:



When this might appear trivial to the experienced user, it might help any unexperienced L<sup>A</sup>T<sub>E</sub>X user (who has never had to describe a programming method, be it for an assignment or not, in a report?). It is also possible to make such a draw using Graphviz, and the dot language. (That depends on your tastes.)

### 1.5.6 A Personalized overline

People sometimes find the result of `\overline` too long, when `\bar`'s one is sometimes too short. It might thus be desirable to have a 'personalized' `\overline` whose length can be adjusted.

#### Code

I found Mr. Els' solution at [\[145\]](#), which needs `amsmath` and `amssymb`:

```

\makeatletter
\newsavebox\myboxA
\newsavebox\myboxB
\newlength\mylenA

\newcommand*\xoverline[2][0.75]{%
  \sbox{\myboxA}{\m@th#2}%
  \setbox\myboxB\null% Phantom box
  \ht\myboxB=\ht\myboxA%
  \dp\myboxB=\dp\myboxA%
  \wd\myboxB=#1\wd\myboxA% Scale phantom
  \sbox\myboxB{\m@th\overline{%
    \copy\myboxB}}% Overlined phantom
  % calc width diff
  \setlength\mylenA{\the\wd\myboxA}
  \addtolength\mylenA{-\the\wd\myboxB}%
  \ifdim\wd\myboxB<\wd\myboxA%

```

```

\rlap{\hskip 0.5\mylenA%
\usebox\myboxB}{\usebox\myboxA}%
\else
\hskip -0.5\mylenA%
\rlap{\usebox\myboxA}{%
\hskip 0.5\mylenA\usebox\myboxB}%
\fi}
\makeatother

```

This defines `\xoverline`, which needs to be called this way:

```
\xoverline[width percent]{symb}
```

You can thus now define a width percent. Two remarks:

1. It will not scale inside sub/super scripts,
2. You can also use it without the `[]` content.

### Example

Consider

$$\overline{W}, \overline{i}, \overline{\overline{i}}, \overline{\mathbb{R}}, \overline{\overline{\mathbb{R}}}, \overline{\overline{\overline{\mathbb{R}}}}$$

which is the output of

```

\xoverline{W}, \xoverline{i}, \xoverline[3.0]{i},
\bar{\mathbb R}, \overline{\mathbb R}, \xoverline{\mathbb R}

```

See [145] for more solutions.

### 1.5.7 Boxes In align-like Environments

At [72], I wondered how I could simply box an equation in an `align`-like environment. Mr. Lars Madsen gave me the solution at [72]. The solution is to use the `calc`, and (evidently) the `amsmath` package too. Then, one can define `\Aboxed` like this:

```

\makeatletter
\newcommand\Aboxed[1]{
% syntax: \Aboxed{ left & right }
\@Aboxed#1\ENDDNE}
\def\@Aboxed#1\ENDDNE{%
% idea: get the left and right part
% typeset them in a \boxed AFTER an ‘&’ and pull it backwards
% but in order to get the horizontal placement to work
% we need to set some appropriate space to the left of the ‘&’
\settowidth\@tempdima{\displaystyle#1}
\setlength\@tempdima{\@tempdima+\fboxsep+\fboxrule}

```

```

% \global does not always mix well with \setlength
\global\@tempdima=\@tempdima
\kern\@tempdima
&
\kern-\@tempdima
\boxed{#1#2}
}
\makeatother

```

This can then be used in the document, simply like this:

```

\begin{align*}
&\boxed{A=B}\\
A&=B\\
&=C
\end{align*}

```

It then gives

$$\begin{array}{c}
 \boxed{A = B} \\
 A = B \\
 = C
 \end{array}$$

Thanks to Mr. Lars Madsen for this trick. One might note that if one wants to box the entire complex then the **empheq** package is a good choice. (Thanks again to Mr. Madsen for this suggestion.)

### 1.5.8 Cases With Square Brackets

At [135], Mr. Kottwitz proposed a solution for drawing cases using square brackets. You only need the **amsmath** package. He took the original **cases** definition of **amsmath**, wrote **sqcases** instead and replaced **\lbrace** by **\lbrack**.

#### Code

```

\makeatletter
\newenvironment{sqcases}{%
  \matrix@check\sqcases\env@sqcases
}{%
  \endarray\right.%
}
\def\env@sqcases{%
  \let\@ifnextchar\new@ifnextchar
  \left\lbrack
  \def\arraystretch{1.2}%

```

```

\array{@{}l@{\quad}l@{}}%
}
\makeatother

```

### Example

The new environment is called `sqcases`, and it gives

$$x_i = \begin{cases} \alpha_i & \text{iff } i > \zeta \\ \alpha_i + \theta & \text{otherwise} \end{cases}.$$

It uses exactly the same syntax as the `cases` environment does. Note that it is also possible (and therefore desirable) to replace the bracket with a line (as `morbusg` did using Plain  $\text{\TeX}$ ).

#### 1.5.9 Closed Square Roots

When writing square roots, some people like to ‘close’ the roots over their content. This way it will look clearer what is inside the square root and what is not. This habit is generally not used while writing with the computer because the text is supposed to be clear anyway, but if you still want to change the output of the square root,  $\text{\LaTeX}$  gives you this possibility. Just add the following code in the preamble:

```

\let\oldsqrt\sqrt
\def\sqrt{\mathpalette\DHLhksqrt}
\def\DHLhksqrt#1#2{%
\setbox0=\hbox{$#1\oldsqrt{#2\,}$}\dimen0=\ht0
\advance\dimen0-0.2\ht0
\setbox2=\hbox{\vrule height\ht0 depth -\dimen0}%
{\box0\lower0.4pt\box2}}

```

This code is only valid for square roots, i.e. you can only use `\sqrt{a}` with this code, not `\sqrt[b]{a}`. You can then check the difference: compare

$$\sqrt{a} \quad \text{to} \quad \sqrt{a}. \quad (1.1)$$

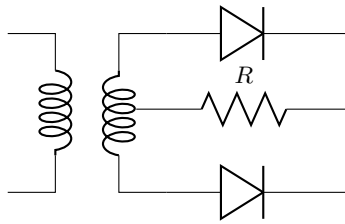
[58]

#### 1.5.10 Correct Circuits

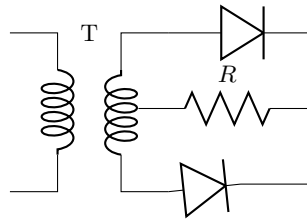
Consider the following circuit, which is basically a redressor.

It is correctly drawn, and compare it with the following one:

The second one looks pretty much the same except for the diode, which is not aligned with the bottom wire of the circuit. Code 2 is



**Fig. 1.1.** Circuit with code 2.



**Fig. 1.2.** Circuit with code 1.

```
\begin{circuitikz} \draw
(0,0) node[transformer] (T) {}
(0.65,-1) to[R=$R$] (3.5,-1)
(T.B2) to[Do] ($(T.B2)+(2,0)$) -| (3.5, -1)
(T.B1) to[Do] ($(T.B1)+(2,0)$) -| (3.5, -1);
\end{circuitikz}
```

when code 1 is

```
\begin{circuitikz} \draw
(0,0) node[transformer] (T) {}
(T.A1) node[anchor=east] {}
(T.A2) node[anchor=east] {}
(T.B1) node[anchor=west] {}
(T.B2) node[anchor=west] {}
(T.base) node{T}
(T.B2) to[Do] (2,-2)
(T.B1) to[Do] (3,0)
(0.65,-1) to[R=$R$] (3,-1)
(2,-2) to[short, -] (3,-2)
(3,-2) to[short, -] (3,0);
\end{circuitikz}
```

Code 2 is an excerpt from what Mr. Redaelli sent me in an e-mail conversation we had. You can clearly see the best way to program in `circuitikz`.

### 1.5.11 Dependency Arrows

For dependency grammar’s teachers, it might be useful to know how to draw dependency arrows between ‘words.’ This method is explained at [10].

#### Code

Consider the phrase

‘The dog eats food.’

Then, first include the `tree-dvips` package in your document’s preamble. Next, declare the nodes:

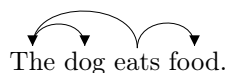
```
\node{The}{The} \node{dog}{dog}%
\node{eats}{eats} \node{food}{food}.
```

Now, you need to draw the arrows:

```
\anodecurve[t]{The}{dog}{1.1em}
\anodecurve[t]{eats}{the}{1.8em}
\anodecurve[t]{eats}{food}{1.2em}
```

#### Result

The result is



### 1.5.12 Diagonally-Cut Cells

To divide a cell by a diagonal, you can use the `slashbox` package. Using the

```
\backslashslashbox{Word 1}{Word 2}
```

command, you will create a cell with ‘Word 1’ and ‘Word 2’ as diagonal elements. [52]

### 1.5.13 Graph Paper

There are two principal alternatives to draw graph paper with L<sup>A</sup>T<sub>E</sub>X.

### Simplest Alternative

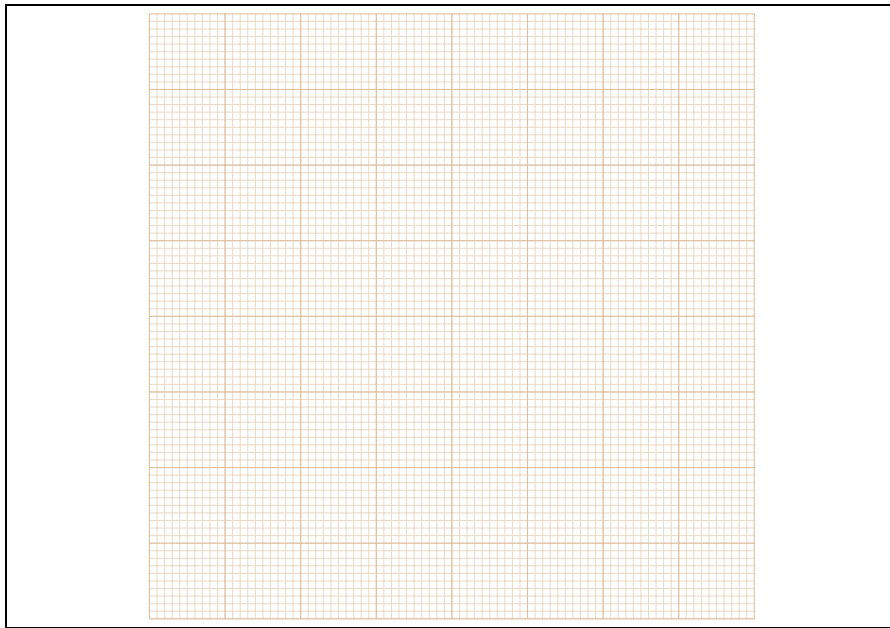
The first one is the simplest, as suggested by Marc van Dongen. Just use

```
\begin{tikzpicture}
\draw[step=0.1cm,very thin,brown!30] (0,0) grid (8,8); \draw[step=1.0cm,thin,brown!50] (0,0) grid (8,8);
\end{tikzpicture}
```

in the body and

```
\usepackage{tikz}
```

in your preamble. You can modify coordinates depending on the size of the graph paper you want. It then results in:



**Trickiest Alternative**

The second solution is Michel Gosse's one. Put

```
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{pst-plot}
\usepackage{multicol}
\usepackage{fancyhdr}

\edef\PstAtCode{\the\catcode'\@}
\catcode'\@=11\relax

\def\psmili{%
  \begingroup%
  \psset{linecolor=bistre}%
  \def\pst@par{}%
  \begin@SpecialObj%
  \pssetxlength\pst@dimg{1cm}
  \pssetylength\pst@dimh{1cm}
  \addto@pscode{%
    /x0 \pst@number\pst@dima \pst@number\pst@dimg div round \pst@number\pst@dimg mul def
    /y0 \pst@number\pst@dimb \pst@number\pst@dimh div round \pst@number\pst@dimh mul def
    /x1 \pst@number\pst@dimc \pst@number\pst@dimg div round \pst@number\pst@dimg mul def
    /y1 \pst@number\pst@dimd \pst@number\pst@dimh div round \pst@number\pst@dimh mul def}
  \addto@pscode{.75 SLW x0 \pst@number\pst@dimg x1 {dup y0 moveto y1 lineto} for stroke}
  \addto@pscode{.3 SLW newpath x0 \pst@number\pst@dimg 2 div x1 {dup y0 moveto y1 lineto} for stroke}
  \addto@pscode{.3 SLW newpath y0 \pst@number\pst@dimh 2 div y1 {dup x0 exch moveto x1 exch lineto} for stroke}
  \addto@pscode{.15 SLW newpath x0 \pst@number\pst@dimh 10 div x1 {dup y0 moveto y1 lineto} for stroke}
  \addto@pscode{.15 SLW newpath y0 \pst@number\pst@dimh 10 div y1 {dup x0 exch moveto x1 exch lineto} for stroke}
  \end@SpecialObj%
  \endgroup}

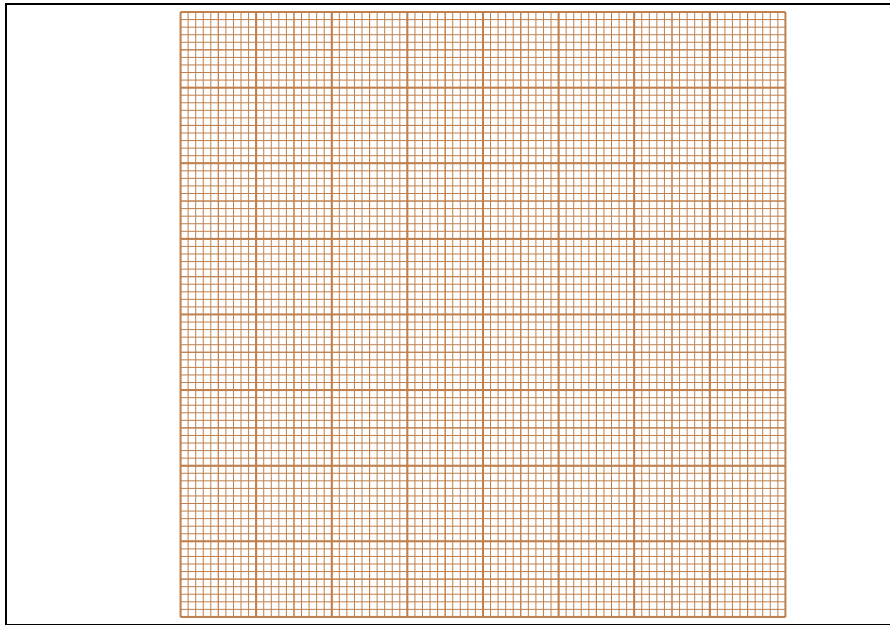
\newrgbcolor{bistre}{.75 .50 .30}
```

in the preamble, and

```
\begin{pspicture}(-4,-4)(4,4)
  \psmili
\end{pspicture}
```

wherever you want (in the body), for your graph paper to appear. You can modify coordinates depending on the size of the graph paper you want.

It then results in:



#### 1.5.14 Logic Gates

*Logic gates* are often used in Electronics and Logic.

##### Independent Drawing

The easiest way to draw logic gates independently is to put

```
\usepackage{tikz}
\usepackage{circuitikz}
```

in the preamble, and to inspire from this:

```
\begin{table}
\begin{center}
\begin{tabular}{||c||c||c||}
\hline
\begin{circuitikz} \draw
(0,0) node[and port] (myand) {}
(myand.in 1) node[anchor=east] {1}
(myand.in 2) node[anchor=east] {2}
(myand.out) node[anchor=west] {3}
;\end{circuitikz} & \begin{circuitikz} \draw
(0,0) node[or port] (myor) {}
(myand.in 1) node[anchor=east] {1}
(myand.in 2) node[anchor=east] {2}
```

```

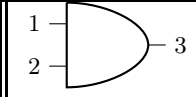
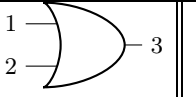
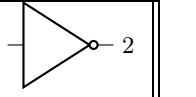
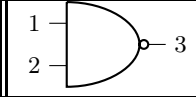
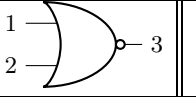
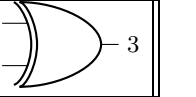
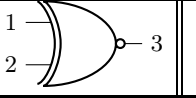
(myand.out) node[anchor=west] {3}
;\end{circuitikz} & \begin{circuitikz} \draw
(0,0) node[not port] (mynot) {}
(mynot.in) node[anchor=east] {1}
(mynot.out) node[anchor=west] {2}
;\end{circuitikz}\\
\hline
AND Door & OR Door & NOT Door\\
\hline
\hline
\begin{circuitikz} \draw
(0,0) node[nand port] (mynand) {}
(mynand.in 1) node[anchor=east] {1}
(mynand.in 2) node[anchor=east] {2}
(mynand.out) node[anchor=west] {3}
;\end{circuitikz} & \begin{circuitikz} \draw
(0,0) node[nor port] (mynor) {}
(mynor.in 1) node[anchor=east] {1}
(mynor.in 2) node[anchor=east] {2}
(mynor.out) node[anchor=west] {3}
;\end{circuitikz} & \begin{circuitikz} \draw
(0,0) node[xor port] (myxor) {}
(myxor.in 1) node[anchor=east] {1}
(myxor.in 2) node[anchor=east] {2}
(myxor.out) node[anchor=west] {3}
;\end{circuitikz}\\
\hline
NAND Door & NOR Door & XOR Door\\
\hline
\hline
& \begin{circuitikz} \draw
(0,0) node[xnor port] (myxnor) {}
(myxnor.in 1) node[anchor=east] {1}
(myxnor.in 2) node[anchor=east] {2}
(myxnor.out) node[anchor=west] {3}
;\end{circuitikz} & \\
\hline
& NXOR Door & \\
\hline
\end{tabular}
\end{center}
\caption{Principal logic gates.}
\end{table}

```

for the body.

### Example

The result is given at Table 1.1, p. 39. Evidently, the other kind of logic gates can be displayed in an analogous way.

		
AND Door	OR Door	NOT Door
		
NAND Door	NOR Door	XOR Door
		
	NXOR Door	

**Table 1.1.** Principal logic gates.

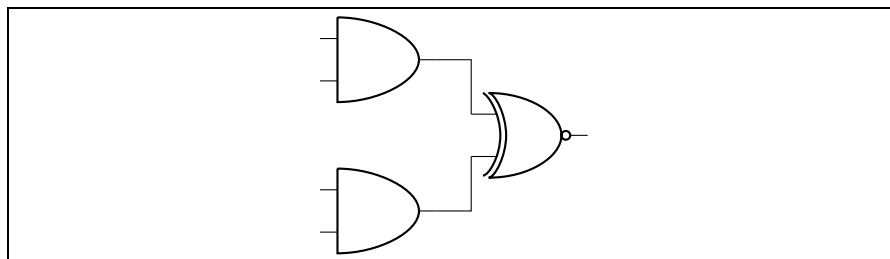
There are other ways to display these gates, such as the “M4 Macros for Electric Circuit Diagrams,” by Dwight Aplevich.

### Linking Them

You can *link them* using different codes. For example, you can use (this example comes from [106])

```
\begin{circuitikz} \draw
(0,2) node[and port] (myand1) {}
(0,0) node[and port] (myand2) {}
(2,1) node[xnor port] (myxnor) {}
(myand1.out) -| (myxnor.in 1)
(myand2.out) -| (myxnor.in 2)
;\end{circuitikz}
```

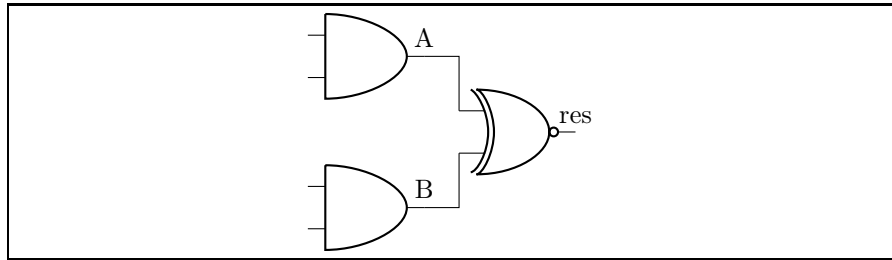
to see



You can also put names, as using

```
\begin{circuitikz} \draw
(0,2) node[and port] (myand1) {}
(0,0) node[and port] (myand2) {}
(2,1) node[xnor port] (myxnor) {}
(myand1.out) node[above] {A} -| (myxnor.in 1)
(myand2.out) node[above] {B} -| (myxnor.in 2)
(myxnor.out) node[above] {res};
\end{circuitikz}
```

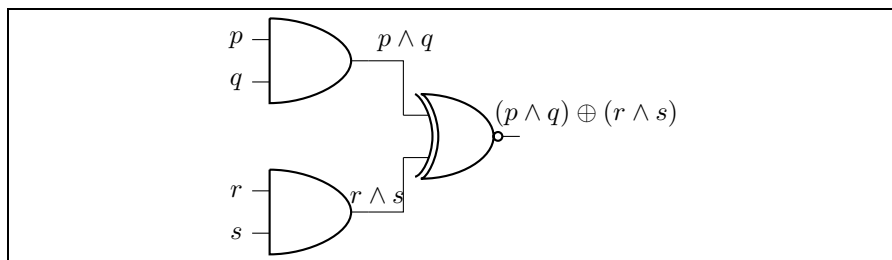
results in (thanks to Massimo Redaelli too for this)



If you want to name entries, you can. Use for example

```
\begin{circuitikz} \draw
(0,2) node[and port] (myand1) {}
(0,0) node[and port] (myand2) {}
(2,1) node[xnor port] (myxnor) {}
(myand1.in 1) node[left] {$p$}
(myand1.in 2) node[left] {$q$}
(myand1.out) node[above right] {$p\wedge q$} -| (myxnor.in 1)
(myand2.in 1) node[left] {$r$}
(myand2.in 2) node[left] {$s$}
(myand2.out) node[above] {$\sim r\wedge s$} -| (myxnor.in 2)
(myxnor.out) node[above] {\quad\quad\quad$(p\wedge q)\oplus (r\wedge s)$}
;\end{circuitikz}
```

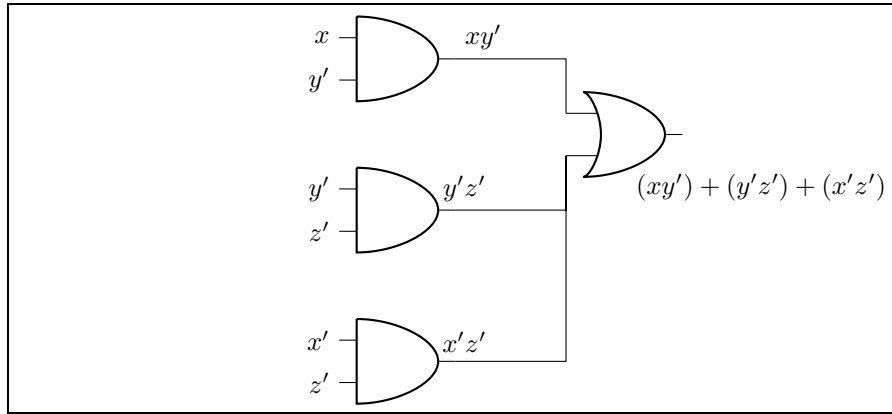
for such an output:



Another example is given by

```
\begin{circuitikz} \draw
(0,2) node[and port] (myand1) {}
(0,0) node[and port] (myand2) {}
(0,-2) node[and port] (myand3) {}
(3,1) node[or port] (myor) {}
(myand1.in 1) node[left] {$x$}
(myand1.in 2) node[left] {$y'$}
(myand1.out) node[above right] {$xy'$} -| (myor.in 1)
(myand2.in 1) node[left] {$y'$}
(myand2.in 2) node[left] {$z'$}
(myand2.out) node[above] {$\sim y' z'$} -| (myor.in 2)
(myand3.in 1) node[left] {$x'$}
(myand3.in 2) node[left] {$z'$}
(myand3.out) node[above] {$\sim x' z'$} -| (myor.in 2)
(myxor.out) node[above] {%
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad%
\quad\quad\quad\quad\quad(xy')+(y'z')+(x'z')} $}
;\end{circuitikz}
```

which gives out



You can do this manually without the lines, but it is really ugly, as using

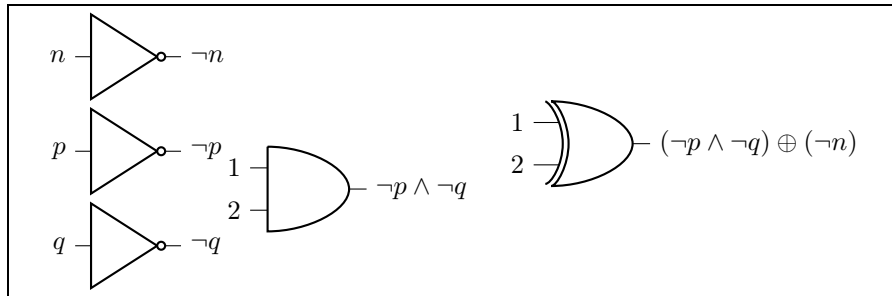
```
\begin{circuitikz} \draw
(0,0) node[not port] (mynot) {}
(mynot.in) node[anchor=east] {$q$}
(mynot.out) node[anchor=west] {$-\neg q$}
(0,1.25) node[not port] (mynot) {}
(mynot.in) node[anchor=east] {$p$}
(mynot.out) node[anchor=west] {$-\neg p$}
(3,0.75) node[and port] (myand) {}
(myand.in 1) node[anchor=east] {1}
(myand.in 2) node[anchor=east] {2}
```

```

(myand.out) node[anchor=west] {\neg p \wedge \neg q}
(6.75,1.35) node[xor port] (myxor) {}
(myxor.in 1) node[anchor=west] {1}
(myxor.in 2) node[anchor=west] {2}
(myxor.out) node[anchor=west] {\neg p \wedge \neg q \oplus (\neg n)}
(0,2.5) node[not port] (mynot) {}
(mynot.in) node[anchor=west] {\neg n}
(mynot.out) node[anchor=west] {\neg n}
;\end{circuitikz}

```

gives



You can also use the `pgflibrarytikzcircuits.code.tex` and `tikzlibrarycircuits.code.tex` files, but the syntax is a little bit different and it becomes really “messy.”

### 1.5.15 Over and Under Braces on Same Elements

It might be desirable to output constructs like

$$a + b + \overbrace{c + d + e + f + g}^x + \underbrace{h + i + k + l}_y = e^2. \quad (1.2)$$

Several solutions were proposed at [142]:

1. You can type the equation twice, once using `\phantom` commands and then raising it:

```

\[a+b+\overbrace{c+d+e+f+g}^x+h+i+k+l=e^2\]
\vspace{-35pt}
\[\phantom{a+b+c+d+}\underbrace{\phantom{e+f+g+h+i}}_y\phantom{+k+=e^2}\]

```

(A similar method is to use `align`, `\hphantom` and `\[size]`.)

2. A short solution is to use `\ooalign` which is defined in L<sup>A</sup>T<sub>E</sub>X’s kernel, and is used in definition of some special text accents and math symbols:

```

\[
\ooalign{

```

```

$a+b+\overbrace{c+d+e+f+g}^{\sim{x}}+h+i+k+l=e^2$\cr
$\phantom{a+b+c+d+{}}\{\underbrace{\phantom{e+f+g+h+i}}_{\sim{y}}\} $\cr
}
\]

```

3. An equally good solution is to use `\rlap`:

```

\[
a+b+\rlap{$\overbrace{\phantom{c+d+e+f+g}}^{\sim{x}}$}c+d+%
\underbrace{e+f+g+h+i}_{\sim{y}}+k+l=e^2
\]

```

4. A total reimplementaion could also define a macro which takes three arguments: the first part only under the overbrace, the middle part between both and the last part only over the underbrace; place this then into a similar `\ialign` structure with three cells and three rows. The braces are then put into the plain $\TeX$  equivalent of `\multicolumn{2}`. See [142] for more details on this.

### 1.5.16 Rightmost Braces

It is common to synthesize some theorems' ideas by using a right brace, or simply to write such kinds of systems to define e.g. a function:

$$\left. \begin{array}{ll} -1 & x \leq 0 \\ 1 & x > 0 \end{array} \right\} \stackrel{\text{def}}{=} f(x). \quad (1.3)$$

This can be achieved by using various tricks, such as those which were proposed by Eduardo Kalinowski and Dan Luecking [51]:

```

♡ \left.
\begin{array}
...\\
... \\
...
\end{array}
\right\}

```

- ♡ Use the `aligned`, `gathered`, or `alignedat` environments,

but one can also

- ♡ define a 'revert cases' environment, say `sesac`, using

```

\usepackage{amsmath}
\makeatletter
\newenvironment{sesac}{%
\let\@ifnextchar\new@ifnextchar
\left.%
\def\arraystretch{1.2}%the default distance between the rows is multiplied by 1.2
% One might prefer aligns other than left,
% depending on the use to which it is put:
\array{@{}l@{\quad}l@{}}%
}{\endarray\right\}}
\makeatother

```

in the preamble, the `amsmath` package evidently being mandatory. One can then use `sesac` the way `cases` is used:

```
\begin{equation}
\begin{cases}
-1 \leq x \leq 0 \\
x > 0
\end{cases} = f(x)
\end{equation}
```

Thanks to Dan Luecking for this [51]. The advantage of this definition is that its call is similar to the one which is used for the `cases` environment. Note that `sesac` will be available through the next `mathtools` update (from June, 2010).

### 1.5.17 Tables With Dashed Lines, Shaded Cells and Arrows Pointing At a Row and a Column

Drawing a table with dashed lines, shaded cells and arrows pointing at a row and a column is possible. You can e.g. use `colortbl`, `multirow`, `bigdelim`, `arydshln` so that [11]

```

\[\begin{array}{c@{ }rc:cccl}
\multicolumn{4}{c}{ } & j & \\
& & & \downarrow & \\
& \ldelim({4}{3.5mm})! ! ! & \% \\
\cellcolor{gray}{1} & \% \\
\cellcolor{gray}{.8} & \% \\
\cellcolor{gray}{.8} & \% \\
\cellcolor{gray}{.8} & \% \\
\!\rdelim{4}{3.5mm}\\
\cdashline{3-6}
& \cellcolor{gray}{.8} & & & \\
i \rightarrow & \% \\
\cellcolor{gray}{.8} & & 1 & \\
& \cellcolor{gray}{.8} & & & \\
\end{array}\]

```

produces

$$i \rightarrow \begin{pmatrix} & \overset{j}{\downarrow} \\ \text{shaded box} & \text{shaded box} \\ \text{shaded box} & 1 \end{pmatrix}$$

## 1.6 Finding

### 1.6.1 Symbols' Commands

The best way to find the command related to a symbol is to have a look in [102]. Searching for your keyword should match your request. You might also try the really interesting tool <http://detexify.kirelabs.org/classify.html>.

## 1.7 Fixing Errors

In a software's development process, the largest part of the time is devoted to code debugging. With L<sup>A</sup>T<sub>E</sub>X, you do not actually create a program, although you write code. There is thus a debugging process. Some beginners deal with errors in a very special way, as they ask "Okay, do these  $n$  errors are noticeable on my document's output? If so, I will have to look at them in a more thorough way. I hope it is not so..." but it is not a good attitude.

If a compiler gives errors, there is a reason: the result (if any!) should be better with no error. Consequently, it is always a good thing to look after the small errors' warnings that `latex` may generate because of your erroneous-prone input. However, saying such things is easy, but errors' causes are sometimes very difficult to find, especially when you write a document of more than 200 pages. As you do when you write programs in C, for example, begin by doing it step-by-step. After much headaches, and, finally, recognizing that you were not really competent before, it is still possible to encounter errors. This time, they won't be easy ones, and you may be unable to find their cause by yourself. That is the aim of this section.

### 1.7.1 Dimen

If you are encountering various weird errors about `dimen` such as

```
! No room for a new \dimen
```

a simple solution is to try

```
\usepackage{etex}
\reserveinserts{100}
```

just after your `\documentclass[...]{...}` command. Usually, the 100 value will be sufficient, and there won't be any errors anymore.

### 1.7.2 Fncychap

If you use

```
\usepackage[Rejne]{fncychap}
```

in your preamble, a code block such as

```
\begin{landscape}
\chapter{A Chapter}
% some text
\end{landscape}
```

will show a bug of the `fncychap` package: the Chapter's box will have two segments of its border moved near the title of the chapter. To solve this issue, just use

```
\let\oldDOTI\DOTI
\renewcommand*\DOTI[1]{\setlength{\textwidth}{\linewidth}\oldDOTI{#1}}
```

after the `\usepackage` declaration. Thanks to Stefan Kottwitz for this trick.

### 1.7.3 L<sup>A</sup>T<sub>E</sub>X

If you encounter very special errors and that you did nothing wrong, use the `fixltx2e` package. There is also a `fix-cm` whose name is self-explanatory. For more pieces of information, please look at [88].

#### `fix-cm`

To use the `fix-cm` package, load it *before* the `\documentclass` command, and use `\RequirePackage` to do so:

```
\RequirePackage{fix-cm}
\documentclass[...]{...}
```

#### `fixltx2e`

To use the `fixltx2e` package, just put

```
\usepackage{fixltx2e}[date]
```

in your preamble, where `date` is the date of the `fixltx2e` package that you are using.

### 1.7.4 PDF compatibility

In our last article, we treated extensively about ‘input encoding.’ Here, we will detail two PDF problems: search and cut-and-past, with their respective solutions. One might generate a PDF document using  $\text{\LaTeX} 2_{\epsilon}$ , but if this document contains accented characters, searching for words containing accented characters could always fail, depending on the PDF reader.

The first problem is that the default font encoding is **OT1**. It should generally deal correctly with basic ASCII characters and the PDF operations on it. But once you will use accented characters, it might cause troubles with some PDF readers.

A solution is to use the **T1** font encoding, conjointly with the **cmap** package. As a result, the

```
% ...
\usepackage[T1]{inputenc}
% ...
\usepackage{cmap}
% ...
```

directive should fix search and cut-and-paste in traditional PDF readers, even for Greek symbols in formulae. [87]

## 1.8 Incorporating

### 1.8.1 MATLAB Graphics

MATLAB can make graphics, but one may want its graphics to be incorporated into a  $\text{\LaTeX}$  document. It can be achieved easily. There are principally two different methods: **laprint** and TikZ/PGF related ones: **Matfig2PGF** and **matlab2tikz**. Thanks to Marc van Dongen for telling me about the second one.

#### LaPrint

For this, you need **laprint.m**. According to [65], once **LaPrint** has been launched into MATLAB, it can perform the following tasks:

- Replace all occurrences of text in the MATLAB figure by tags,
- Save the modified figure in Postscript format (**eps** file),
- Create a **tex** file which contains commands of the  $\text{\LaTeX}$  **psfrag** package to replace the tags by the original text and to call the postscript file.

Let’s assume you have typed

```
>> set(0,'defaulttextinterpreter','none')
>> figure(1),clf
>> plot([1 2])
>> ylabel('A straight line')
```

where “>>” denotes MATLAB’s prompt. Let’s then type (assuming `laprint.m` is in your current MATLAB working directory)

```
>> laprint
```

**LaPrint** thus asks you the “Number of Figure to be saved” and “Basename of Files to be Created.” You can modify several options, then click on “Go!.” The `laprint` script will thus create two files: an `eps` one and a `tex` one.

The `tex` file can be included into  $\text{\LaTeX}$  documents using the packages `graphicx`, `color` and `psfrag`. Thus, if you let “Basename of Files to be Created” to “unnamed,” a simple `tex` file showing your graphics will be generated, and will have such a content:

```
\documentclass{article}
\usepackage{graphicx,color,psfrag}
\begin{document}

\input{unnamed}

\end{document}
```

This is very easy. For other pieces of information (such as how to give a predetermined size to your graphics, ...), do not hesitate to read [65].

## Matfig2PGF

**Matfig2PGF** converts a MATLAB figure to the Portable Graphics Format (PGF). This PGF file can be included in a  $\text{\LaTeX}$  document. Once `matfig2pgf` has been launched into MATLAB, you just need to generate your plot in MATLAB, and then, invoke `matfig2pgf` using

```
>> matfig2pgf('myfile.pgf')
```

where >> is MATLAB’s prompt and `myfile.pgf` is the output file. You can now write your `.tex` document according to the following minimal structure:

```
\documentclass{article}

\usepackage{pgf}
\usepackage{pgffor}
\usepgflibrary{plohandlers}

% Or, for older PGF versions (<= 1.01)
%\usepackage{pgf}
```

```

%\usepackage{pgffor}
%\usepackage{pgflibraryplohandlers}

\begin{document}
  \begin{figure}
    \centering
    \input{myfile.pgf}
    \caption{Figure created by Matfig2PGF}
  \end{figure}
\end{document}

```

This is a really easy way to put a MATLAB figure into a  $\text{\LaTeX}$  document. For more pieces of information, simply type

```
>> help matfig2pgf
```

in MATLAB.

### Matlab2Tikz

An easier way to achieve this is to use `matlab2tikz`. Once `matlab2tikz` has been launched into MATLAB, you just need to generate your plot in MATLAB, and then, invoke `matlab2tikz` using

```
>> matlab2tikz('myfile.tikz');
```

where `>>` is MATLAB's prompt and `myfile.tikz` is the output file. You can now write your `.tex` document according to the following minimal structure:

```

\documentclass{article}
\usepackage{tikz}
\usepackage{pgfplots}
\begin{document}
  \input{myfile.tikz}
\end{document}

```

This is a really easy way to put a MATLAB figure into a  $\text{\LaTeX}$  document. For more pieces of information, please have a look at [1]. You may note that you can do all these things by simply using Sage $\text{\TeX}$ , but it is a little bit less straightforward. You may note that you can do all these things by simply using Sage $\text{\TeX}$ , but it is a little bit less straightforward. It is also possible that you might have to use more than one approach (especially coupling `laprint` with the two other approaches). If you were, for example, to plot the result of a `spectrogram` command in MATLAB, the only way to include it in a  $\text{\LaTeX}$  document is to use `laprint`.

### 1.8.2 Spreadsheets

It is sometimes desirable to *export* some *spreadsheets into L<sup>A</sup>T<sub>E</sub>X*. It can be useful for many purposes, such as scientific experiments (collected data, for example), or simply for financial reports. It is easily achieved thanks to `Calc2LaTeX`.

### 1.8.3 References

We have seen before (Subsection 1.14.19, p. 70) that you can use `\ref{yourreference}` to give the reference of `yourreference`. If your document has a lot of pages, using references such as `mytable1` and such ones won't be practical and useful. These references are here to help you. The same remark can be done for the bibliography (whatever your use of BiB<sub>T</sub>E<sub>X</sub>), but it is up to you to decide about a convention on your bibliography labels.

Since this, it is better to use a personal grammar to generate them. For example, a widespread solution is to write `sec:nameofthesection` for the section called `nameofthesection`, and so on for other parts. For an equation, you could use `eq:myequation`. The most important thing is to give coherent names to labels.

## 1.9 Modifying

### 1.9.1 Captions

Some journals which do not have a specific class ask for small captions. It can be achieved easily using

```
\usepackage{caption}
\captionsetup{figurename=Figure, tablename=Table}

\renewcommand{\captionfont}{\small}
\renewcommand{\captionlabelfont}{\bfseries}
```

With this code, you are also able to modify the name of two L<sup>A</sup>T<sub>E</sub>X floats: `figure` and `table`.

### 1.9.2 Margins and Marginpar Notes

This is a note in the margin.

Let's say you want to make `\marginpar{}` notes like the one in this margin, but that you have modified your margins. You have to modify your `\marginpar{}` command. If you do not do this, it can result in a bad layout, depending on the amount of space you added, or took, from your previous margins. An example of a code to correct this is given by

```

\let\mymarginpar\marginpar
\marginparwidth=2.55cm
\marginparsep=3.2pt
\renewcommand\marginpar[1]{%
  \mymarginpar{\raggedright\hbadness=10000\tiny\it #1\par}}

```

if you have previously defined margins like this:

```

\addtolength{\voffset}{-1.5cm}
\addtolength{\textheight}{2cm}
\addtolength{\hoffset}{-2cm}
\addtolength{\textwidth}{4.2cm}

```

It can be easily adapted to the configuration you want to work with. The configuration given here is quite interesting, as using this does not make your document looking “compressed,” but helps you to return the space on the pages. Note that, according to [175], you can use other measure units than cm. They are classed by order of importance (*i.e.* the first is the most used one in L<sup>A</sup>T<sub>E</sub>X, the last the less used):

1. **pt**: a point is  $\frac{1}{72.27}$  inch, that means about 0,0138 inch or 0,3515 mm,
2. **bp**: a big point is  $\frac{1}{72}$  inch, that means about 0,0139 inch or 0,3527 mm,
3. **mm**: a millimeter,
4. **in**: inch,
5. **ex**: roughly the height of an “x” in the current font,
6. **em**: roughly the width of an “M.”

There are also the length macros’ parts:

1. **\baselineskip**: the normal vertical distance between lines in a paragraph,
2. **\baselinestretch**: to multiply **\baselineskip**,
3. **\columnsep**: the distance between columns,
4. **\columnwidth**: the width of the column
5. **\evensidemargin**: the margin for “even” pages (think of a printed booklet),
6. **\linewidth**: the width of a line in the local environment,
7. **\oddsidemargin**: the margin for “odd” pages (think of a printed booklet),
8. **\paperwidth**: the width of the page,
9. **\paperheight**: the height of the page,
10. **\parindent**: the normal paragraph indentation,
11. **\parskip**: the extra vertical space between paragraphs,
12. **\tabcolsep**: the default separation between columns in a tabular environment,
13. **\textheight**: the height of text on the page,
14. **\textwidth**: the width of the text on the page,
15. **\topmargin**: the size of the top margin.

### 1.9.3 The Zoom of a Document

To magnify a document, you may simply write `\mag=xxxx`, where `xxxx` may be 1440, for example. It must be put before the `\documentclass` command!

## 1.10 Printing

### 1.10.1 Empty Pages

If you need to create an artificial blank page, you need to put something into it so that  $\text{\LaTeX}$  takes it into account. For this, you can e.g. use [4]

```
% text
\newpage
\mbox{}
\newpage
```

This might be interesting if you need to meet some specific criteria for e.g. a report.

### 1.10.2 Monochrome

When writing a ‘screen-version’ of some book, one often uses colors. However, if this screen-version needs to be printed in black and white, it is better to give it as a monochrome document. This can be achieved easily by simply adding `monochrome` to `color` and `xcolor`’s options. For example, if you called these packages without any options, it means that you might put

```
\usepackage[monochrome]{color}
\usepackage[monochrome]{xcolor}
```

in your preamble. Thanks to Enrico Gregorio [171] for this.

Herbert Voß gave me [171] a more technical and PS version:

```
\AtBeginDocument{%
\special{ps:
  /setcmykcolor { exch 0.11 mul add
                  exch 0.59 mul add
                  exch 0.3 mul add
                  dup 1 gt { pop 1 } if neg 1 add setgray } def
  /setrgbcolor { 0.11 mul
                  exch 0.59 mul add
                  exch 0.3 mul add setgray } def
  /sethsbcolor { /b exch def /s exch def 6 mul dup cvi dup /i exch def
sub /f exch def
                  /F [[0 1 f sub 1][f 0 1][1 0 1 f sub][1 f 0][1 f sub
1 0][0 1 f][0 1 1]] def
```

```

F i get { s mul neg 1 add b mul} forall
0.11 mul
exch 0.59 mul add
exch 0.3 mul add setgray } def
}
}

```

Thanks to him too.

## 1.11 Sizing

### 1.11.1 Boxed Minipages Depending On the Text

Putting

```

\usepackage{fancybox}
\usepackage{boxedminipage}
\usepackage{lipsum}

```

in the preamble, and

```

\begin{boxedminipage}{\textwidth}
\lipsum[n]
\end{boxedminipage}

```

in the body, results in this, for  $n = 5$ :

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

If you want other boxes, check the **fancybox** package's documentation. Please do not use these boxes in an excessive way: it would result in an ugly layout, and, worse, a confused reader.

### 1.11.2 `\left` and `\right` to Produce Smaller Brackets

If you want `\left` and `\right` to produce smaller (big) brackets, you can define a `\left-\right` command, say `\E`, using

```

\usepackage{amssbsy}
\newcommand{\E}[1]{%
  \setbox8\hbox{${\#1}$}%
  \newdimen\tempheight
  \tempheight=\ht8
  \advance\tempheight-.2em
  \:\boldsymbol{E}\!
  \left(\vbox to \tempheight{}\right.
  \kern-.25em #1\kern-.25em
  \left.\vbox to \tempheight{}\right)
}

```

You can then use e.g. `\E{}` and put what needs to be parenthesized between the brackets [11]. Compare the result:

$$\boldsymbol{E}\left(\int f(x) \, dx\right) \quad \text{to} \quad \boldsymbol{E}\left(\int f(x) \, dx\right).$$

This is an example where a bold E is printed with, but this might evidently be implemented without this!

## 1.12 Splitting

### 1.12.1 algorithm Environments on Multiple Pages

It is generally impossible to split `algorithm` environments on multiple pages. As a result, you can define a new `algorithmbis` environment using [11]

```

\usepackage{algorithmic}

\makeatletter
\newcounter{algorithmbis}
\setcounter{algorithmbis}{0}
\renewcommand{\thealgorithmbis}{%
% \thesection.\arabic{algorithmbis}}
\def\algorithmbis{\@ifnextchar[%
% {\@algorithmbisa}\@algorithmbisb}}
\def\@algorithmbisa[#1]{%
  \refstepcounter{algorithmbis}
  \trivlist
  \leftmargin\z@
  \itemindent\z@
  \labelsep\z@
  \item[\parbox{\textwidth}{%
    \hrule
    \hrule
  }%
}

```

```

\noindent\strut\textbf{Algorithm%
\thealgorithmbis} #1
\hrule
}] \hfil\vskip0em%
}
\def\endalgorithmbis{%
\hfil\vskip-1em\hrule\endtrivlist}
\makeatother

```

so that you can now use

```

\begin{algorithmbis}[Name]\label{myalgorithm}
\begin{algorithmic}
% ...
\end{algorithmic}
\end{algorithmbis}

```

More details are available at [11].

## 1.13 Using

### 1.13.1 \centering and the center Environment Accordingly

In [81], I explained how to avoid erroneous references for floats. I said that the `center` environment might have been replaced by the `\centering` command, but did not explain the differences between those two concepts.

A detailed explication has been given at [147] by `egreg`.

1. `\centering` needs a `\par` (or an empty line) before the closing brace. It is usually used inside some environment that provides the necessary `\par`, such as `minipage` or `figure`.
2. The main difference is that `\centering` does not leave vertical space before and after it (`\begin{center}` is defined in terms of `trivlist`).

It is thereby generally preferred to use `\centering` over the `center` environment, e.g. in `figure` environments.

### 1.13.2 \scalebox

In many situations, `\scalebox` might be useful. Briefly, you can use

```

\scalebox{factor}{%
% some thing to scale
}

```

where what you want to be scaled might be an image, a `tabular` environment, etc. This is the same kind of command as the well-known `\resizebox`, which takes a fixed size (width and, potentially, height). Both require the `\graphicx` package. [185]

### 1.13.3 The Enumerate Package

For some purposes, one would for example want such a list:

```
[1]: My first item,
[2]: My second item.
```

It can be achieved easily using

```
\usepackage{enumerate}
```

in the preamble, and

```
\begin{enumerate}[[1]:]
\item My first item,
\item My second item.
\end{enumerate}
```

in the body of the document. Note that the body's code can be replaced with other codes such as

```
\begin{enumerate}[[1]{:}]
\item My first item,
\item My second item.
\end{enumerate}
```

but the first code block was the most obvious (and clear) way to do this.

## 1.14 Writing

### 1.14.1 Aligned Systems of (In)Equations

Systems of aligned (in)equations were generally typeset using tricky combinations of environments. I recently discovered the **systeme** package by Christian Tellechea. This package allows you to typeset e.g.

$$\begin{cases} 2a - 3b + 4c = 2 \\ a + 8b + 5c = 8 \\ -a + 2b + c = -5 \end{cases}$$

using

```
\systeme{2a-3b+4c=2,a+8b+5c=8,-a+2b+c=-5}
```

Consider having a look at [\[155\]](#) if you are interested.

### 1.14.2 (a) Bold `\ell`

There are briefly (at least; [143]) two ways to write a bold  $\ell$ :

1. Using `amsmath` and `amsbsy`, you can define

```
\newcommand*\Bell{\ensuremath{\boldsymbol{\ell}}}
```

so that `\Bell` produces the desired effect,

2. Without any packages, you can use `\boldmath` and `\unboldmath`:

```
\boldmath$\ell$\unboldmath
```

(that you can define in a command `\Bell` too, if desired).

### 1.14.3 A Symmetric Matrix

I saw at [125] three different approaches to write a symmetric matrix. All were given by Mr. Donig.

Using only `amsmath`,

```
\[
\begin{bmatrix}
a&b&c\\
&d & {0} \\
\multicolumn{2}{c}{\text{\smash{\raisebox{%
1.5ex}{Sym.}}}} & {0}
\end{bmatrix}
```

produces

$$\begin{bmatrix} a & b & c \\ \text{Sym.} & d & 0 \\ & 0 & 0 \end{bmatrix}.$$

Using the `multirow` package in supplement to the `amsmath` package, you can also write

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ & 1 & 1 & 1 \\ & & 1 & 1 \\ \text{Sym.} & & & 1 \end{bmatrix}$$

using

```
\begin{bmatrix}
1 & 1 & 1 & 1 \\
& 1 & 1 & 1 \\
& & 1 & 1 \\
& & & 1
\end{bmatrix}
```

or

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ & 1 & 1 & 1 \\ & & 1 & 1 \\ \text{Sym.} & & & 1 \end{bmatrix}$$

using

```
\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
& 1 & 1 & 1 \\
& & 1 & 1 \\
& & & \text{Sym.} & 1
\end{bmatrix}
```

#### 1.14.4 Code Depending On the Processor

According to [175], as your documents get more complicated, it could be necessary to make some changes in the source so that it will work for an output format but it will not for the other. For example, all that is related to graphics has to be adapted according to the final format. Even if you should use different pictures according to the final format, you can override this limit putting in the same folder pictures in different formats (e.g. EPS and PNG) with the same name and link them without writing the extension. There is a simple way to solve this problem:

```
\usepackage{ifpdf}
```

in the preamble, or, if you don't have this package, you can add the following text just after `\documentclass[...]{...}`:

```
\newif\ifpdf
\ifx\pdfoutput\undefined
  \pdffalse
\else
  \ifnum\pdfoutput=1
    \pdftrue
  \else
    \pdffalse
  \fi
\fi
```

This is plain TeX code. The `ifpdf` package and this code both define a new if-else that you can use to change your code according to the compiler you are using. After having used this code, you can use whenever you want in your document the following syntax:

```

\ifpdf
  % we are running pdflatex
\else
  % we are running latex
\fi

```

For example, you can use this syntax to load different packages according to the compiler.

If you can do without this, it is better. For example, including PNG images for PDF documents, when you are using PDF<sub>T</sub>E<sub>X</sub>, works, but making a conversion to EPS or PS format is always nicer. Richard Socher modified a serie of files of the **Sam2p** software, written by Szabó Péter. You are now able to transform any (suitable) JPG or PNG file into a EPS file. This works under Microsoft Windows, but you can manage to make it work on Linux. This is really handy. I invite you to use **Inkscape** as a really interesting program to transform images between formats. It has a **Texttext** plugin which allows you to put any block of L<sup>A</sup>T<sub>E</sub>X code in your image.

### 1.14.5 Dancing Text

I am not aware of any L<sup>A</sup>T<sub>E</sub>X easter eggs, and a search on the Internet lead me to [131], where Mr. Le Floch proposes some code to write dancing text. I adapted his version to only make **\emph**-declared text as dancing. The dancing angle is set between  $-10^\circ$  and  $10^\circ$ .

Consider the following working example.

```

\documentclass{article}
\usepackage{rotating}
\usepackage[first=-10,last=10]{lcg}
\makeatletter
\newcommand{\globalrand}{\rand\global%
\cr@nd\cr@nd}
\makeatother
\newcommand{\randompos}[1]{%
  \expandafter\let\csname old\string#1%
  \endcsname#1%
  \expandafter\def\expandafter#1%
  \expandafter##\expandafter1\expandafter{%
    \csname old\string#1\endcsname{\protect%
      \globalrand\protect\turnbox{%
        \value{rand}}{##1}\protect%
        \phantom{##1}}}%
  }
\randompos\emph
\begin{document}

```

```

\section{Example}
\emph{Some} \emph{emphasized} \emph{text},
\emph{with} \emph{random} \emph{directions}

\end{document}

```

You can evidently declare e.g. `\randompos\section` if you want the sections' titles to be dancing too. For example,

```

\dancing{Here} \dancing{is}
\dancing{an}\dancing{example!}

```

produces

*Here is an example!*

Note that there are evidently other techniques to produce dancing text.

### 1.14.6 Dash Integrals

You can use the standard `\strokedint` from `MnSymbol` [103], which draws an oblique line on an integral, but you can also define, according to [117],

```

\def\Xint#1{\mathchoice
{\XXint\displaystyle\textstyle{#1}}%
{\XXint\textstyle\scriptstyle{#1}}%
{\XXint\scriptstyle\scriptscriptstyle{#1}}%
{\XXint\scriptscriptstyle\scriptscriptstyle{#1}}%
\!\int}
\def\XXint#1#2#3{{\setbox0=\hbox{#1{#2#3}}%
\int}$ }
\vcenter{\hbox{#2#3$ }}\kern-.6\wd0}}
\def\ddashint{\Xint=}
\def\dashint{\Xint-}

```

which results in

$$f, \int, \mathrel{\int}, \mathrel{\int}$$

for the successive

```

dashint, \displaystyle\dashint, %
\ddashint, \displaystyle\ddashint

```

calls.

### 1.14.7 Dashed And Underlined Text

Putting

```

\usepackage{ulem}
\def\dotuline{\bgroup
  \ifdim\ULdepth=\maxdimen % Set depth based on font, if not set already
    \settodepth\ULdepth{(j)\advance\ULdepth.4pt\fi
  \markoverwith{\begingroup
    \advance\ULdepth0.08ex
    \lower\ULdepth\hbox{\kern.15em .\kern.1em}}%
  \endgroup}\ULon}

\def\dashuline{\bgroup
  \ifdim\ULdepth=\maxdimen % Set depth based on font, if not set already
    \settodepth\ULdepth{(j)\advance\ULdepth.4pt\fi
  \markoverwith{\kern.15em
    \vtop{\kern\ULdepth \hrule width .3em}}%
  \kern.15em}\ULon}

```

in the preamble, and

```
\dotuline{This sentence is dotted}, \dashuline{This sentence is dashed}
```

in the body results in:

<p><u>Th</u><u>is</u><u> s</u><u>en</u><u>te</u><u>nc</u><u>e</u><u> i</u><u>s</u><u> d</u><u>ott</u><u>e</u><u>d</u><u>,</u><u> </u><u>Th</u><u>is</u><u> s</u><u>en</u><u>te</u><u>nc</u><u>e</u><u> i</u><u>s</u><u> d</u><u>as</u><u>he</u><u>d</u><u>.</u></p>
---

You can also use my `dashundergaps` package. For this, please check [79].

#### 1.14.8 Date and Time

It is sometimes useful to write the date and time at compilation in a document. This can be achieved using the `datetime` package, together with

```

\shortdayofweekname{\day}{\month}{\year}
\shortmonthname{} \twodigit{\day}
\hhmmsstime{}
TIMEZONE \number\year

```

where ‘TIMEZONE’ is your time zone. [129]

Note that metadata serve this role too, and that this does not replace metadata. However, this might be useful if you have a tendency to modify often some specific documents. By this method, people always know if they have the last version, or not.

#### 1.14.9 Diagonal (Inverse) Dots

Dots are well-known:

- `\vdots` produces ∴,

- `\cdots` produces  $\cdots$ ,
- `\ldots` produces  $\ldots$ ,
- `\ddots` produces  $\ddots$ .

However, for strange reasons, you might need to use an inverse-diagonal version of `\ddots` such as  $\cdot^{\cdot^{\cdot}}$ . This can be achieved using `\iddots` from the `mathdots` package [11].

#### 1.14.10 Different Slides

If you do not like the traditional slides you can make with traditional options, such as these ones:

```
\documentclass[a4paper]{slides}
\begin{document}
\begin{slide}
First text in the first slide.
\end{slide}
\begin{slide}
First text in the second slide.
\end{slide}
\end{document}
```

you can use

```
\documentclass[16pt,landscape]{article}
\usepackage{geometry}
\geometry{paperheight=297mm,paperwidth=223mm}
\usepackage{sectsty}
\sectionfont{\Huge}
\subsectionfont{\LARGE}
\subsubsectionfont{\Large}
\paragraphfont{\large\it}
\subparagraphfont{\normalsize\it}

\addtolength{\voffset}{-1.5cm}
\addtolength{\textheight}{2cm}
\addtolength{\hoffset}{-2cm}
\addtolength{\textwidth}{4.2cm}
```

in the preamble, and

```
\begin{landscape}
%\maketitle
\tableofcontents
HELLO.
\Huge test
```

```

\section{First Section}
\subsection{First Subsection}
\subsubsection{First Subsubsection}
\paragraph{First Paragraph}
\subparagraph{First Subparagraph}
\begin{theorem}[First Theorem]
Iff  $a+b=0$ ,  $a=-b$ .
\end{theorem}
\end{landscape}

```

in the body.

#### 1.14.11 Enumerations With Textcircled Numbers

Using the `enumerate` package, you can write

```

\begin{enumerate}[\textcircled{\arabic{enumi}}]
\item Item 1
\item Item 2
\item \ldots
\item Item  $n$ 
\end{enumerate}

```

for such a result:

- ① Item 1
- ② Item 2
- ③ ...
- ④ Item  $n$

This is simple to achieve, but it might improve some boring enumerated lists. Do not forget to put

```
\usepackage{enumerate}
```

in the preamble.

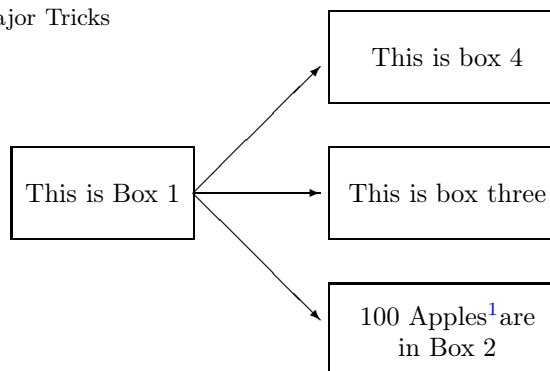
#### 1.14.12 Footnotes In Boxes

##### Example

Say you want this.

---

<sup>1</sup> This is a fruit



You will notice that a footnote was placed in the bottom of the page, as desired. Generally, generating footnotes from a box is not that easy because of internal `\footnote` limitations. [96] You might use a different syntax than `\put` and `\framebox` for boxes, but if you want to stick with this syntax, a first solution is to use `\footnotemark` that generates the footnote marking in the text, together with `\footnotetext{...}` that generates the footnote text. The former needs to be put in the box (so that the number actually appears in the box), when the latter needs to be given outside the box, but on the same page. [96]

### Code

I used

```
\footnotetext{This is a fruit}
\addtocounter{footnote}{-1}

{\setlength{\unitlength}{6mm}
\begin{picture}(13,7)(0,0)

\put(0,3){\framebox(4,2){This is Box 1}}

\put(7,0){\framebox(5,2){
\shortstack{100 Apples\footnotemark are\\%
in Box 2}}}
\put(4,4){\vector(1,1){2.8}}

\put(7,3){\framebox(5,2){
\shortstack{This is box three}}}
\put(4,4){\vector(1,0){2.8}}

\put(7,6){\framebox(5,2){
\shortstack{ This is box 4}}}
\put(4,4){\vector(1,-1){2.8}}
```

```
\end{picture}
}
```

The advantage of using this approach is that it is relatively ‘clean,’ i.e. you are not obliged to use `\textsuperscript`, be it in the box, or in the footnote, for numbering. Thanks to Heiko Oberdiek at [96] for this.

#### 1.14.13 Footnotes In tabular

You can use footnotes in a `tabular` environment. Consider for example the following (output of a `tabular` environment:

word1 <sup>1</sup>	word2 <sup>2</sup>
--------------------	--------------------

This does not prevent you from using `\footnote` elsewhere.

This is achieved using [11]

```
\begin{tabular}{|c|c|}
\hline
word1\footnotemark[1] & word2\footnotemark[2] \\
\hline
\end{tabular}
\footnotetext[1]{Note for word1.}
\footnotetext[2]{Note for word2.}
```

assuming you are using the `footnote` package together with the

```
\makesavenoteenv{tabular}
```

option. More details are provided at [11].

#### 1.14.14 Footnotes With a New Number At Each Section

You can use two different alternatives to use footnotes with a new numbering at each page. The easiest one is to put

```
\@addtoreset{footnote}{section}
```

in the preamble of your document. Another one is to put

```
\setcounter{footnote}{0}
```

before every section. Note that `section` can be replaced by another counter, such as one of those detailed at Section 1.3, p. 14. This is a practical example of how to implement a counter’s resetting with an already-defined counter.

---

<sup>1</sup> Note for word1.

<sup>2</sup> Note for word2.

### 1.14.15 Footnotes Without Any Number

If you need to make a footnote without any number, you can use

```
\makeatletter
\newcommand{\ack}[1]{\let\save@makefntext\@makefntext%
  \def\@makefntext##1{\parindent0em##1}\footnotetext{#1}%
  \let\@makefntext\save@makefntext}
\makeatother
```

in the preamble, and

```
\ack{This work was supported by the \textit{fund} Fund.}
```

thus resulting in the footnote you can see on the bottom of this page. Thanks to Justus Piater for this trick.

### 1.14.16 Fractions

The basic Math commands are well known, and will not be cited. However, some authors use commands which are not well-known. For example, compare

$$\frac{1}{k} \log_2 c(f)$$

to

$$\frac{1}{k} \log_2 c(f).$$

It is clear that the second fraction is smaller than the first one. This is simply because `\tfrac` means “adapt `\frac` to text-style.” There is also `\dfraction`, which means “adapt `\frac` to display-style.” For other interesting details, you may check [13]. Please do not use `\over` and other  $\TeX$  primitives!

### 1.14.17 Indented Text, But Not the First Line

Let us consider that you want to indent some text, but not the first line of the text.

For this, you can use the following structure [11]:

```
\noindent Test
\begin{list}{}{\settowidth\labelwidth{}
  \setlength{\leftmargin}{1em}
  \addtolength{\itemindent}{-1em}}
\item This is some sentence that might [...]
\item This is some sentence that might [...]
\end{list}
\noindent Test
```

---

This work was supported by the *fund* Fund (this is false if there exists such a fund).

### 1.14.18 Integrals Adapted to the Context

Let's say you need to show that you integrate both sides of a system of ordinary differential equations (ODEs). Typically, it would be interesting to put integrals before matrices, but ... what if the matrices are too big? Your integrals will appear as tiny compared to these matrices. The same problem arises if you integrate a big fraction, but in this case, the fraction might be reduced before, except if it is shown like this for pedagogical reasons.

We shall here develop the integrals for different matrices' sizes. This is easily adapted to fractions. Matrices' sizes will be considered from the biggest one you could expect to the smallest one. Evidently, even the smallest one will contain a fraction, to make it "too big." Thanks to pg for his related trick, in the message on the website

<http://www.les-mathematiques.net/phorum/read.php?10,472951>.

#### Matrices With Five Rows

Compare

$$\int_{t_i}^{t_f} \begin{pmatrix} \frac{S(1-\alpha)-\sigma_{\text{eff}}T^4-k_W\frac{dW_s}{dt}}{k} \\ P-mC_s \\ -P+mC_s+e(t) \\ -H+E \\ H-E \end{pmatrix} dt \quad \text{to} \quad \int_{t_i}^{t_f} \begin{pmatrix} \frac{S(1-\alpha)-\sigma_{\text{eff}}T^4-k_W\frac{dW_s}{dt}}{k} \\ P-mC_s \\ -P+mC_s+e(t) \\ -H+E \\ H-E \end{pmatrix} dt.$$

To achieve

$$\int_{t_i}^{t_f} \begin{pmatrix} \frac{S(1-\alpha)-\sigma_{\text{eff}}T^4-k_W\frac{dW_s}{dt}}{k} \\ P-mC_s \\ -P+mC_s+e(t) \\ -H+E \\ H-E \end{pmatrix} dt$$

you simply need to use

```
\makeatletter
\newcommand{\bigint}{\@ifnextchar_{\@bigintsub\@bigintnosub}
\def\@bigintsub_{#1}{\def\@int@subscript{#1}\@ifnextchar^{\@bigintsubsup\@bigintnosub}
\def\@bigintsubsup^#1{\mathop{\text{\Huge$\int_{\text{\normalsize$\scriptstyle\kern-0.35em%
\@int@subscript$}}\text{\normalsize$\scriptstyle#1$}}}\nolimits}
\def\@bigintsubnosup{\mathop{\text{\Huge$\int_{\text{\normalsize$\scriptstyle\@int@subscript$}}}\nolimits}
\def\@bigintnosub{\@ifnextchar^{\@bigintnosubsup\@bigintnosubnosup}
\def\@bigintnosubsup^#1{\mathop{\text{\Huge$\int^{\text{\normalsize$\scriptstyle#1$}}}\nolimits}
\def\@bigintnosubnosup{\mathop{\text{\Huge$\int}\nolimits}
\makeatother
```

in your preamble, and then use `\bigint` at the place of `\int` before the matrix.

### Matrices With Four Rows

Compare

$$\int_{t_i}^{t_f} \begin{pmatrix} \frac{S(1-\alpha)-\sigma_{\text{eff}}T^4-k_W\frac{dW_s}{dt}}{k} \\ P-mC_s \\ -P+mC_s+e(t) \\ -H+E \end{pmatrix} dt \quad \text{to} \quad \int_{t_i}^{t_f} \begin{pmatrix} \frac{S(1-\alpha)-\sigma_{\text{eff}}T^4-k_W\frac{dW_s}{dt}}{k} \\ P-mC_s \\ -P+mC_s+e(t) \\ -H+E \end{pmatrix} dt.$$

To achieve

$$\int_{t_i}^{t_f} \begin{pmatrix} \frac{S(1-\alpha)-\sigma_{\text{eff}}T^4-k_W\frac{dW_s}{dt}}{k} \\ P-mC_s \\ -P+mC_s+e(t) \\ -H+E \end{pmatrix} dt$$

you simply need to use

```
\makeatletter
\newcommand{\bigints}{\@ifnextchar_{\@bigintssub\@bigintsnosub}
\def\@bigintssub_{#1}{\def\@int@subscript{#1}\@ifnextchar^{\@bigintssubsup\@bigintssubnosup}
\def\@bigintssubsup^#1{\mathop{\text{\huge$\int_{\text{\normalsize$\scriptstyle\@int@subscript$}}$}}\nolimits}
\def\@bigintssubnosup{\mathop{\text{\huge$\int_{\text{\normalsize$\scriptstyle\@int@subscript$}}$}}\nolimits}
\def\@bigintsnosub{\@ifnextchar^{\@bigintsnosubsup\@bigintsnosubnosup}
\def\@bigintsnosubsup^#1{\mathop{\text{\huge$\int^{\text{\normalsize$\scriptstyle\@int@subscript$}}$}}\nolimits}
\def\@bigintsnosubnosup{\mathop{\text{\huge$\int$}}\nolimits}
\makeatother
```

in your preamble, and then use `\bigints` at the place of `\int` before the matrix.

### Matrices With Three Rows

Compare

$$\int_{t_i}^{t_f} \begin{pmatrix} \frac{S(1-\alpha)-\sigma_{\text{eff}}T^4-k_W\frac{dW_s}{dt}}{k} \\ P-mC_s \\ -P+mC_s+e(t) \end{pmatrix} dt \quad \text{to} \quad \int_{t_i}^{t_f} \begin{pmatrix} \frac{S(1-\alpha)-\sigma_{\text{eff}}T^4-k_W\frac{dW_s}{dt}}{k} \\ P-mC_s \\ -P+mC_s+e(t) \end{pmatrix} dt.$$

To achieve

$$\int_{t_i}^{t_f} \begin{pmatrix} \frac{S(1-\alpha)-\sigma_{\text{eff}}T^4-k_W\frac{dW_s}{dt}}{k} \\ P-mC_s \\ -P+mC_s+e(t) \end{pmatrix} dt$$

you simply need to use

```

\makeatletter
\newcommand{\bigintss}{\@ifnextchar_{\@bigintsssub\@bigintssnosub}
\def\@bigintsssub_{#1}{\def\@int@subscript{#1}\@ifnextchar^{\@bigintsssubsup\@bigintsssubnosup}
\def\@bigintsssubsup^#1{\mathop{\text{\LARGE$\int_{\text{\normalsize$\scriptstyle\kern-0.25em%
\@int@subscript$}}\text{\normalsize$\scriptstyle#1$}}}\nolimits}
\def\@bigintsssubnosup{\mathop{\text{\LARGE$\int_{\text{\normalsize$\scriptstyle\@int@subscript$}}}\nolimits}
\def\@bigintssnosub{\@ifnextchar^{\@bigintssnosubsup\@bigintssnosubnosup}
\def\@bigintssnosubsup^#1{\mathop{\text{\LARGE$\int^{\text{\normalsize$\scriptstyle#1$}}}\nolimits}
\def\@bigintssnosubnosup{\mathop{\text{\LARGE$\int$}}\nolimits}
\makeatother

```

in your preamble, and then use `\bigintss` at the place of `\int` before the matrix.

## Matrices With Two Rows

Compare

$$\int_{t_i}^{t_f} \left( \frac{S(1-\alpha) - \sigma_{\text{eff}} T^4 - k_W \frac{dW_s}{dt}}{P - mC_s} \right) dt \quad \text{to} \quad \int_{t_i}^{t_f} \left( \frac{S(1-\alpha) - \sigma_{\text{eff}} T^4 - k_W \frac{dW_s}{dt}}{P - mC_s} \right) dt.$$

To achieve

$$\int_{t_i}^{t_f} \left( \frac{S(1-\alpha) - \sigma_{\text{eff}} T^4 - k_W \frac{dW_s}{dt}}{P - mC_s} \right) dt$$

you simply need to use

```

\makeatletter
\newcommand{\bigintsss}{\@ifnextchar_{\@bigintsssub\@bigintssnosub}
\def\@bigintsssub_{#1}{\def\@int@subscript{#1}\@ifnextchar^{\@bigintsssubsup\@bigintsssubnosup}
\def\@bigintsssubsup^#1{\mathop{\text{\LARGE$\int_{\text{\normalsize$\scriptstyle\kern-0.20em%
\@int@subscript$}}\text{\normalsize$\scriptstyle#1$}}}\nolimits}
\def\@bigintsssubnosup{\mathop{\text{\LARGE$\int_{\text{\normalsize$\scriptstyle\@int@subscript$}}}\nolimits}
\def\@bigintssnosub{\@ifnextchar^{\@bigintssnosubsup\@bigintssnosubnosup}
\def\@bigintssnosubsup^#1{\mathop{\text{\LARGE$\int^{\text{\normalsize$\scriptstyle#1$}}}\nolimits}
\def\@bigintssnosubnosup{\mathop{\text{\LARGE$\int$}}\nolimits}
\makeatother

```

in your preamble, and then use `\bigintsss` at the place of `\int` before the matrix.

## Matrices With One Row

Compare

$$\int_{t_i}^{t_f} \left( \frac{S(1-\alpha) - \sigma_{\text{eff}} T^4 - k_W \frac{dW_s}{dt}}{k} \right) dt \quad \text{to} \quad \int_{t_i}^{t_f} \left( \frac{S(1-\alpha) - \sigma_{\text{eff}} T^4 - k_W \frac{dW_s}{dt}}{k} \right) dt.$$

To achieve

$$\int_{t_i}^{t_f} \left( \frac{S(1-\alpha) - \sigma_{\text{eff}} T^4 - k_W \frac{dW_s}{dt}}{k} \right) dt$$

you simply need to use

```
\makeatletter
\newcommand{\bigintssss}{\ifnextchar_\@bigintsssssub\@bigintssssnosub}
\def\@bigintsssssub_#1{\def\@int@subscript{#1}\ifnextchar^\@bigintsssssubsup\@bigintsssssubnosup}
\def\@bigintsssssubsup^#1{\mathop{\text{\large$\int_{\text{\normalsize$\scriptstyle\kern-0.15em%
\@int@subscript$}}\text{\normalsize$\scriptstyle#1$}}}\nolimits}
\def\@bigintsssssubnosup{\mathop{\text{\large$\int_{\text{\normalsize$\scriptstyle\@int@subscript$}}\text{\normalsize$\scriptstyle#1$}}}\nolimits}
\def\@bigintssssnosub{\ifnextchar^\@bigintssssnosubsup\@bigintssssnosubnosup}
\def\@bigintssssnosubsup^#1{\mathop{\text{\large$\int^{\text{\normalsize$\scriptstyle#1$}}\text{\normalsize$\scriptstyle#1$}}}\nolimits}
\def\@bigintssssnosubnosup{\mathop{\text{\large$\int^{\text{\normalsize$\scriptstyle#1$}}}\nolimits}
\makeatother
```

in your preamble, and then use `\bigintssss` at the place of `\int` before the matrix. This is here a matter of taste, as both symbols are typographically acceptable.

### The Bigints Package

I have developed the `bigints` package which allows you to use these five kinds of integrals. You can thus forget the code, and simply use

```
\usepackage{bigints}
```

in your preamble, and then choose between `\bigint`, `\bigints`, `\bigintss`, `\bigintsss`, `\bigintsssss`.

Consider the manual of my `bitints` package if you are interested: [82].

#### 1.14.19 Links

`Hyperref`'s package's options have clear names which are directly related to their respective roles. It may be difficult in `hyperref`'s manual to find the most important options one could use. For this reason, a traditional way to use `hyperref` is given here:

```
\usepackage[a4paper,bookmarks=true,bookmarksnumbered=true,bookmarksopen=true,
bookmarksopenlevel=1,breaklinks=true,colorlinks=true,final,menucolor=red,
pdfauthor={The Author},pdfcreator={The Creator (should be an application)},
pdfkeywords={first keyword, second one, ...},pdftitle={The Title of The PDF},
pdfsubject={The Subject},pdftoolbar=true]{hyperref}
\hypersetup{urlcolor=red,linkcolor=blue,citecolor=blue,colorlinks=true}
```

From now on, you will be able to write a URL with

```
\url{http://www.myurl.mydomain}
```

giving “<http://www.myurl.mydomain>” or even with

```
\href{http://www.myurl.mydomain}{My URL}
```

giving “**My URL**” and make an *internally-linked* reference with

```
\ref{yourlabel}
```

for the reference’s number, and

```
\pageref{yourlabel}
```

for the reference’s page, such as “see how to count rows in tables at the Point [1.3.4](#), p. 16.”

With the PDF options we declared, PDF tags are automatically modified according to the options you passed to the `hyperref` package.

### Avoiding URLs In the Margin

If there is a URL which comes (at least) partly in the margin, because it does not want to split, it is normal. By default, `hyperref` does not split URLs. To solve this issue, just invoke `breakurl`:

```
\usepackage{breakurl}
```

after the `hyperref`’s block. Your URLs will now be splitted if needed. It gives

```
http://www.myurl.mydomain/myfirstfolder/mysecondfolder/
mythirdone/myfourthone/thenameofmyautomaticallygeneratedpage.
php?firstoption1=0&firstoption2=0&firstoption3=42
```

#### 1.14.20 Logic Inferences

To typeset

$$\frac{\frac{[(p \rightarrow \perp) \rightarrow \perp] \quad [p \rightarrow \perp]}{\perp} (RAA)}{((p \rightarrow \perp) \rightarrow \perp) \rightarrow p} (\rightarrow I)$$

you might consider using the `proof` package using e.g.

```
\infer[(\rightarrow I)]
{((p\rightarrow \perp) \rightarrow \perp) \rightarrow p}
{\infer[(RAA)]
{p}
{\infer[(\rightarrow E)]
{\perp}
{[(p\rightarrow \perp) \rightarrow \perp] \rightarrow p}
& [p\rightarrow \perp] }}
}
```

Thanks to Mr. Tennent for this idea [156]. There are obviously other packages to typeset inferences.

## 1.14.21 Matrices

## Surrounded by...

The most used way to typeset matrices is

```
\begin{pmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{pmatrix}
```

It results in

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

Another way is to write

```
\begin{vmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{vmatrix}
```

to have

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

which often denotes  $\det A$ , if  $A$  is composed by the  $a_{ij}$ ,  $1 \leq i \leq 3$ ,  $1 \leq j \leq 3$ .

It is even possible to write

$$\left\| \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \right\|$$

This can be achieved using

```
\begin{Vmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{Vmatrix}
```

Brackets are often used too:

$$\left[ \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \right]$$

done with

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

An old-fashioned (and misunderstood) way to typeset matrices is to use

```
\begin{Bmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{Bmatrix}
```

which shows

$$\left\{ \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \right\}$$

## With Braces

For pedagogical reasons, one can want such a result:

$$\left( \begin{array}{c|c} a & \overbrace{b \dots c}^{n \text{ times}} \\ \hline d & \\ \vdots & A \\ e & \end{array} \right)$$

This can be achieved using different codes which have to be put in the preamble of one's document. These codes come from [11].

A first solution would be to use

```
\usepackage{multirow}
\makeatletter
\def\Biggg#1{{\hbox{$\left#1\ vbox to32\p@{}\right.\n@space$}}}
\newdimen\bracketwidth
\settowidth{\bracketwidth}{\Biggg{}}
\makeatother
\[
\begin{array}{r@{\hspace{\arraycolsep}}rcc%
c@{\hspace{\arraycolsep}}c@{1}
&&&\multicolumn{3}{c}{\vspace{-.5em}
\overbrace{\hphantom{b \hspace{2\arraycolsep} \cdots
\hspace{2\arraycolsep} c}}^{n \mbox{\scriptsize times}}}\}
\multirow{5}{\bracketwidth}[3pt]{\Biggg{}}
&&a&\multirow{5}{1pt}[3pt]{\vrule height 52pt}&b&\cdots&c&
\multirow{5}{\bracketwidth}[3pt]{\Biggg{}}
\cline{2-7}
&&d\\
&&\vdots&&&A\\
&&e
\end{array}
\]
```

Another is to use

```
\[
\ vbox{%
\hskip2.8em$\overbrace{\hphantom{b \hspace{2\arraycolsep} \cdots
\hspace{2\arraycolsep} c}}^{n \mbox{\scriptsize times}}}$
\vskip-.25em
$\left(
\begin{array}{r|ccc}
a & b & \cdots & c
\end{array}
\right)
```

```

\hline
d\\
\vdots & & A\\
e
\end{array}
\right)$
}
\]

```

Another is

```

\def\moverbrace#1#2{%
\newdimen\moverbracewd%
\settowidth\moverbracewd{#1}%
\addtolength\moverbracewd{-2\arraycolsep}
\vbox to 1.6ex{\hsize=\moverbracewd\centering\vss
$\overbrace{#1}^{\#2}$}%
}
\[
\left(
\begin{array}{r|c}
a & \moverbrace{b \hspace{2\arraycolsep} \cdots \\
\hspace{2\arraycolsep} c}{n \ \mbox{\scriptsize times}} \\
\hline
d\\
\vdots & A\\
e
\end{array}
\right)
\]

```

A last one is

```

\def\moverbrace#1#2{%
\newdimen\moverbracewd%
\settowidth\moverbracewd{#1}%
\vbox to 1.6ex{\hsize=\moverbracewd\centering\vss
$\overbrace{#1}^{\#2}$}%
}
\[
\left(
\begin{array}{c|c}
a & \moverbrace{b \hspace{2\arraycolsep} \cdots \\
\hspace{2\arraycolsep} c}{n \ \mbox{\scriptsize times}} \\
\hline
d\\
\vdots & A
\end{array}
\right)

```

```

      e
    \end{array}
  \right)
\]

```

We saw

```

\makeatletter
% commands with ‘‘at’’
\makeatother

```

in some of these codes. That is because the “at” character is a special one for L<sup>A</sup>T<sub>E</sub>X. It cannot be directly used in a macro.

### With Borders

When you have such a system:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= y_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= y_2 \\ \vdots &= \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= y_n \end{cases},$$

you sometimes want to write its *associated matrix*, i.e.

$$\left( \begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & y_1 \\ a_{21} & a_{22} & \vdots & a_{2n} & y_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & y_n \end{array} \right)$$

To write this associated matrix, just use

```

\left(
  \begin{array}{cccc|c}
    a_{11}&a_{12}&\cdots&a_{1n}&y_1\\
    a_{21}&a_{22}&\vdots&a_{2n}&y_2\\
    \vdots&\vdots&\ddots&\vdots&\vdots\\
    a_{n1}&a_{n2}&\cdots&a_{nn}&y_n
  \end{array}
\right)

```

**With Full Borders**

To write such a matrix:

$$\left(\begin{array}{c|c} a & 0 \\ \hline 0 & 0 \end{array}\right)$$

it is wise to write it too in an `array` environment. As every `tabular`, you can use `\hline` to draw a horizontal line. You then have

```
\left(
  \begin{array}{c|c}
    a&0\\
\hline
    0&0
  \end{array}
\right)
```

Evidently, you can adapt this to use `{}`, and other constructs.

**With Dots**

Let us consider that you want to write a matrix with vertical dots between two columns. It is achieved easily using

```
\left(
\begin{array}{@{}cc@{}c@{}c@{}}
a & b & & c\\
d & e & & f\\
g & h & \makebox[2\arraycolsep]{\smash{\vdots}} & i
\end{array}
\right)
```

in a `math` environment, or with the low-level equivalent:

```
\left(
\begin{array}{@{}cc@{}c@{}c@{}}
a & b & & c\\
d & e & & f\\
g & h & \hbox to 2\arraycolsep{\hss\smash{\vdots}\hss} & i
\end{array}
\right)
```

(still in a `math` environment). Thanks to Philipp Stephani for this trick.

### With Text

Putting text along with a matrix, and thus wanting

$$\begin{array}{cccc} & a_1 & a_2 & \cdots & a_n \\ b_1 & \left( \begin{array}{cccc} 1.2 & 3.3 & 5.1 & 2.8 \\ 4.7 & 7.8 & 2.4 & 1.9 \\ \cdots & \cdots & \cdots & \cdots \\ 8.0 & 9.9 & 0.9 & 9.99 \end{array} \right) \\ c_1 & & & & \\ \cdots & & & & \\ z_1 & & & & \end{array}$$

can appear to be impossible. However, the following code (from [11]):

```
\bordermatrix{{& a_1 & a_2 & \cdots & a_n & \cr
               b_1 & 1.2 & 3.3 & 5.1 & 2.8 & \cr
               c_1 & 4.7 & 7.8 & 2.4 & 1.9 & \cr
               \cdots & \cdots & \cdots & \cdots & \cdots & \cr
               z_1 & 8.0 & 9.9 & 0.9 & 9.99 & \cr}}
```

works pretty well. It can be easily adapted for other purposes. The difference between `\\` and `\cr` is that `\\` cannot be used in `\bordermatrix` because this plain  $\text{\TeX}$  command uses the base, primitive, commands of the language, and `\\` is not part of them; `\cr` may be used in `tabular` in substitution of `\\`, because it is a primitive command and because it has a unique and not ambiguous meaning, while `\\` may mean different things, besides being a fragile command because it admits several optional arguments, the asterisk and the bracket delimited vertical spacing; sometimes `\cr` solves these ambiguities, but both can be used in plain  $\text{\TeX}$  and in  $\text{\LaTeX}$  in order to terminate an alignment line even before its last column, without considering, therefore, the useless trailing ampersands.

### With Same Space Between Elements When `\boxed` is Used

Let us consider the following (Sudoku) matrix:

$$\begin{pmatrix} \boxed{6} & 7 & 3 & 5 & 1 & 4 & 9 & 8 & 2 \\ 9 & \boxed{2} & 1 & 3 & 6 & 8 & 7 & 5 & 4 \\ 5 & 8 & \boxed{4} & 7 & 9 & 2 & 1 & 3 & 6 \\ 8 & 9 & 6 & 2 & 3 & 7 & 4 & 1 & 5 \\ 2 & 1 & 5 & 9 & 4 & 6 & 3 & 7 & 8 \\ 3 & 4 & 7 & 1 & 8 & 5 & 2 & 6 & 9 \\ 4 & 5 & 2 & 8 & 7 & 1 & 6 & 9 & 3 \\ 1 & 6 & 9 & 4 & 5 & 3 & 8 & 2 & 7 \\ 7 & 3 & 8 & 6 & 2 & 9 & 5 & 4 & 1 \end{pmatrix}.$$

How do you manage to make the spacing the same between columns, even for columns which do not have any `\boxed` element? There are various solutions which were proposed by Mr. Voß and ‘GL’ at [27]:

1. You might use the `tabu` environment (with `X`-columns),
2. You might redefine a `\Boxed` command using the `mathtools` package together with

```
\def\Boxed#1{\mathclap{\fboxsep=1pt\boxed{#1}}}
```

Using the second method, one gets directly

$$\left(\begin{array}{cccccccc} \boxed{6} & 7 & 3 & 5 & 1 & 4 & 9 & 8 & 2 \\ 9 & \boxed{2} & 1 & 3 & 6 & 8 & 7 & 5 & 4 \\ 5 & 8 & \boxed{4} & 7 & 9 & 2 & 1 & 3 & 6 \\ 8 & 9 & 6 & 2 & 3 & 7 & 4 & 1 & 5 \\ 2 & 1 & 5 & 9 & 4 & 6 & 3 & 7 & 8 \\ 3 & 4 & 7 & 1 & 8 & 5 & 2 & 6 & 9 \\ 4 & 5 & 2 & 8 & 7 & 1 & 6 & 9 & 3 \\ 1 & 6 & 9 & 4 & 5 & 3 & 8 & 2 & 7 \\ 7 & 3 & 8 & 6 & 2 & 9 & 5 & 4 & 1 \end{array}\right)$$

by replacing `\boxed` elements with `\Boxed` ones. If you decide to put e.g. bigger numbers (such as 60 or 200), you might think about increasing `\arraycolsep` (e.g. to 15pt).

### 1.14.22 Minitocs

*Minitocs* are table of contents (TOC's) for a given hierarchy's item. They are very useful in one's document, especially if the related items' hierarchy becomes difficult to understand or to keep with, because of much text between them.

A very easy way to set minitocs just after the title of a `\chapter{...}`, for example, is to use

```
\usepackage{minitoc}
\setcounter{minitocdepth}{1}
\mtcselectlanguage{english}
\setlength{\mtcindent}{24pt} % default setting
\renewcommand{\mtcfont}{\small\rmfamily\upshape\mdseries} % default setting
\renewcommand{\mtcSfont}{\small\rmfamily\upshape\bfseries} % default setting
```

in the preamble,

```
\dominitoc
```

just after the

```
\begin{document}
```

and then

```
\chapter{...}
\minitoc
```

each time you want it to appear. We can parameter independently these variables:

- `mtcindent`: indentation (left/right) of minitocs,
- `mtcfont`: font for the minitoc,
- `mtcSfont`: font for the minitoc (sections),
- `mtcSSfont`: font for the minitoc (subsections),
- `mtcSSSfont`: font for the minitoc (subsubsections),
- `mtcPfont`: font for the minitoc (paragraphs),
- `mtcSPfont`: font for the minitoc (subparagraphs),
- `mlffont`: font for the minilof (figures),
- `mltfont`: font for the minilot (tables),
- `mtifont`: fonts for titles.

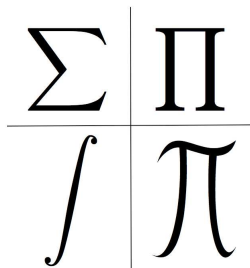
Note that, in this document, there is a minitoc before the beginning of each Chapter, but we used

```
\setcounter{minitocdepth}{3}
```

so that `\section{name}`, `\subsection{name}` and `\subsubsection{name}` are displayed too.

### 1.14.23 Product Integrals

It might happen for you to need to draw ‘product integrals.’ Consider having a look at [9] for this (Richard D. Gill’s home page). You can then draw the product integral symbol, as illustrated in the right bottom corner of Figure 1.3.



**Fig. 1.3.** Four mathematical symbols, the one at the right bottom corner being defined as the ‘product integral’ symbol.

### 1.14.24 Roman Numbers

As shown at [172], defining

```

\makeatletter
\newcommand{\rmnum}[1]{\romannumeral #1}
\newcommand{\Rmnum}[1]{\expandafter%
  \@slowromancap\romannumeral #1@}
\makeatother

```

in the preamble lets you use `\rmnum{num}` to typeset `num` as a lowercase roman numeral, and `\Rmnum{num}` to typeset it as an uppercase roman numeral.

### 1.14.25 Simplifications

According to [109], if you put

```
\usepackage{cancel}
```

in your preamble, and

```
\dfrac{\cancel{(b-a)} (a^2+ab+b^2)}{\cancel{(b-a)} (b+a)}
```

in the body, it results in

$\frac{\cancel{(b-a)}(a^2 + ab + b^2)}{\cancel{(b-a)}(b + a)}$
--

### 1.14.26 Small mathcal Letters

When you give a lower-case letter as the argument of `\mathcal`, you generally end up with a non-letter symbol, and not a small calligraphic letter. However, `\mathcal` is known to produce calligraphic letters when its argument is an upper-case letter.

I found an explanation of this fact at [128]. I here give it, somewhat remodified. T<sub>E</sub>X was created long before character sets and fonts were standardized by Unicode and OpenType. Each mathematical font contains only 128 glyphs (“newer” versions of T<sub>E</sub>X allow 256 glyphs, but the Computer Modern math fonts use only 128).

For Latin letters, the default font is `cmmi` (Computer Modern Math Italic) which contains uppercase and lowercase letters; the `\mathcal` command simply switches all letters to `cmsy` (Computer Modern Math Symbols), which happens to contain calligraphic uppercase letters at the correct positions, but no lowercase calligraphic letters; instead it has various operators and other symbols (thus the font name ‘Symbols’) at the position where normally the lowercase letters reside. You can find the exact encoding tables in the L<sup>A</sup>T<sub>E</sub>X Encoding Guide (`texdoc encguide`), section A.4.

This behavior is unrelated to the AMS fonts; the AMS fonts do not contain any medium-weight calligraphic characters.

### 1.14.27 Stacks (In Equations)

#### First Kind

If you make some mathematics, you know that some authors write things such as

$$f(x) \stackrel{\text{def}}{=} \ln(1 + e^{-x}) + \frac{x}{3}$$

but you always wondered how they can do this. In fact, it is not very difficult to put something on the = operator:

`\stackrel{\text{\tiny def}}{=}`

produces  $\stackrel{\text{def}}{=}$ .

#### Second Kind

One could write

$$\begin{aligned} \int \frac{dx}{e^x - 1} &= \frac{x}{e^x - 1} + \int \frac{x \cdot e^x}{(e^x - 1)^2} dx \quad \left| \begin{array}{l} u = e^x - 1 \\ \Leftrightarrow du = e^x \\ \text{and } x = \ln(u + 1) \end{array} \right. \\ &= \frac{x}{e^x - 1} + \int \frac{\ln(u + 1) du}{u^2}. \end{aligned}$$

This is achieved using

```
\begin{align*}
\int \frac{\mathrm{d}x}{e^x-1} &= \frac{x}{e^x-1} + \int \frac{x \cdot e^x}{(e^x-1)^2} \mathrm{d}x \quad \left| \begin{array}{l} u = e^x - 1 \\ \Leftrightarrow du = e^x \\ \text{and } x = \ln(u+1) \end{array} \right. \\
&= \frac{x}{e^x-1} + \int \frac{\ln(u+1)}{u^2} du.
\end{align*}
```

The important concept is evidently the

```
\shortstack[c]{u=e^x-1} \Leftrightarrow du=e^x \\
\text{and } x=\ln(u+1)
```

which gives, in its context,

$$\left. dx \right| \begin{array}{l} u = e^x - 1 \\ \Leftrightarrow du = e^x \\ \text{and } x = \ln(u + 1) \end{array}$$

According to [109], you can also use a different code. For example, to display

$$\lim_{\substack{x \rightarrow 1 \\ x > 1}} \frac{1+x^2}{x-1} = +\infty$$

you can use

```
\lim_{\{x\to 1\}\atop{x>1}} \}\dfrac{1+x^2}{x-1}=+\infty
```

Among other possibilities, it is also possible to define a `\stack` command so

that you can write

$$\lim_{x_2 \succ x_{0,2}} \mathcal{T}([x_{0,1}, x_2], [x_{0,1}, x_2]) = x_0,$$

$$\sum_{\substack{i,j \in \{1,2,\dots,n\} \\ i < j}} f_{ij}$$

using [11]

```
\displaystyle \lim_{x_2 \stackrel{\to}{>} x_{0,2}}
\mathcal{T}([x_{0,1}, x_2], [x_{0,1}, x_2]) = x_0$
\[
\sum_{\stackrel{i,j}{\in \{1,2,\ldots,n\}}{i < j}} f_{ij}
\]
```

and having defined

```
\newcommand{\stack}[2]{%
\genfrac{}{}{0pt}{}{#1}{#2}}
```

#### 1.14.28 Standard Display of Date and Time

It is sometimes useful to write the date and time at compilation in a document. This can be achieved using the `datetime` package, together with

```
\shortdayofweekname{\day}{\month}{\year}
\shortmonthname{} \twodigit{\day} \hhmmss{time}
TIMEZONE \number\year
```

where ‘TIMEZONE’ is your time zone. [129] Note that metadata serve this role too.

#### 1.14.29 Subequations with Braces

One could want such a result:

$$\begin{cases} a_1(x) = b_1 & (1.4a) \\ a_2(x) = b_2 & (1.4b) \\ a_3(x) = b_3. & (1.4c) \end{cases}$$

This is achieved easily thanks to the inclusion of the `empheq` package, with the following code:

```

\begin{subequations}
\begin{empheq}[left=\empheqlbrace]{align}
a_1(x) &= b_1\\
a_2(x) &= b_2\\
a_3(x) &= b_3.
\end{empheq}
\end{subequations}

```

Thanks to Mr. Heller for this trick, at [35].

### 1.14.30 Tables

*Tables* are a natural way to display (judiciously chosen) data without taking too much space.

#### With Braces

If you want to make a table with a brace into it, just put

```

\usepackage{multirow}
\usepackage{bigdelim}

```

in the preamble, and

```

\begin{tabular}{l|l|r}
\hline
$a$ & $b$ & \\
\hline
$c$ & $d$ & \multirow{2}{*}{\rdelim\}{2}{1cm}[text]\\
$e$ & $f$ & 
\end{tabular}

```

in the body.

It results in

$a$	$b$	
$c$	$d$	}text
$e$	$f$	

### 1.14.31 Tabular Packages: What are Their Respective Purposes, and Which Ones Conflict?

When looking through different topics on StackExchange, I found an extremely interesting question: [148]. There are effectively many `tabular` environments, and it is desirable to find a quick summary on

- i. the different roles and limitations of the packages,
- ii. the potential conflicts between them.

Mr. Alan Munn, Stefan Kottwitz, ‘lockstep’ and ‘Altermundus’ gave a quick, yet quite comprehensive answer to these two questions.

## Respective Purposes

They distinguished basic packages over different categories:

- **Basic packages**
  - `array` offers more flexible column formatting, and fixes some spacing issues. This is an almost ‘must-use’ package,
  - `booktabs` supports professional looking tables, creates a better vertical spacing, better rules, and is specifically designed for tables without vertical lines (the norm for publication-quality tables),
  - `cellspace` ensures a minimal spacing of table cells,
  - `dcolumn` creates columns which align on a decimal marker. There are similar packages such as `numprint`, `rccol`, `warpcol` (not to forget `siunitx` whose role is to format correctly quantities depending on their unit, be it in the SI or not; it also provides the `S` column type that aligns on a decimal mark),
  - `multirow` lets tabular material span multiple rows,
  - `tabularx` provides a column type which expands to fill the specified width of the table,
  - `tabulary` provides column types which are proportional to the natural width of their contents;
- **Multi-page tables**
  - `longtable` provides tabulars that can split across pages,
  - `ltablex` also combines features of `longtable` and `tabularx`,
  - `ltxtable` combines features of `longtable` and `tabularx`,
  - `supertabular` provides tabulars that can split across multiple pages,
  - `xtab` extends `supertabular` by some features and improves page breaking;
- **Captioning and notes**
  - `floatrow` (although mainly about customizing layouts of float environments) allows for footnotes and additional explanations in tables,
  - `threeparttable` typesets tables with captions and notes matching width,
  - `threeparttablex` provides the functionality of `threeparttable` to tables created using `longtable`;
- **Color and fancy features**
  - `arydshln` prints horizontal and vertical dashed lines,
  - `bigdelim` inserts variable-sized multi-row delimiters into a table,
  - `blkarray` splits arrays into blocks and add delimiters for each block,
  - `colortab` shades and colors tables,
  - `colortbl` colors rows/columns/cells. The `xcolor` package with the `table` option provides alternating table row colors to extend it,
  - `delarray` adds delimiters (braces, parentheses, brackets) to arrays,
  - `hhline` allows better double line producing,
  - `makecell` allows multiple line cells, better headers, gaps in cells, numbered rows, thick lines, diagonally divided cells, etc.,

- `slashbox` allows diagonally divided tabular cell;
- **All-rounder**
  - `tabu` is a single package that provides much of the functionality of many of the above packages.

They also especially recommend four packages:

1. `array`, because it is a universal talent for tuning whole columns by commands,
2. `booktabs` is a ‘must’ for professional-looking layout,
3. `longtable` is very popular for multi-page tables,
4. `tabularx` is great for auto-sizing columns.

Other packages are then used depending on the required features.

### Packages Conflicts

There are some well-known conflicts between some of these packages:

- According to the manual, `arydshln` has to be loaded after `array`, `longtable`, `colortab`, and `colortbl`, respectively,
- `xtab` and `supertabular` do not work together. Loading both would cause a conflict. `xtab` should be preferred, since it is written as an extension to `supertabular`.

#### 1.14.32 Text On the Same Line as an Equation

To achieve

Let us consider  $x \mapsto ax + b = f(x)$

(that is, to put some text before an equation), you might consider using

```
\[
\makebox[\displaywidth]{%
  \makebox[0pt][l]{Let us consider}\hfill
  $\displaystyle x\mapsto ax+b=f(x)$\hfill}
\]
```

Note that

- the procedure needs to be somewhat modified if the equation needs to be numbered,
- it is extremely bad style.

Thanks to the contributors at [30].

Any Text that you  
want above ...



...or below the im-  
age.

**Fig. 1.4.** A caption

### 1.14.33 Text Below an Image

One would for example want a text output such as Figure 1.4, p. 86. This can be achieved easily using the following code, from [7].

```
\begin{figure}[!ht]
  \centering
  \parbox{0.25\linewidth}{%
    Any Text that you want above \ldots\smallskipamount]
    \rule{\linewidth}{0.25\linewidth}\smallskipamount]
    \ldots or below the image.
  }
  \caption{A caption}\label{fig:label1}
\end{figure}
```

### 1.14.34 Two Itemizations on the Same Line

Say you want

1. First item
2. Second item
3. – 4. Third and fourth items
5. Fifth item

This is achieved thanks to

```
\begin{enumerate}
\item First item
\item Second item
\item[\refstepcounter{enumi}\labelenumi\ --%
\refstepcounter{enumi}\labelenumi] Third and fourth items
\item Fifth item
\end{enumerate}
```

Thanks to Philipp Stephani (see [152]).

### 1.14.35 Verbatim in Center

Using a `verbatim` environment in a `center` environment will not put the `verbatim` environment's content in the center of the page. This can be achieved, e.g. using the `fancyvrb` package. For this, consider e.g. using `BVerbatim` environment in a `center` environment [11]:

```
\begin{center}
\begin{BVerbatim}[boxwidth=auto]
Test test
\end{BVerbatim}
\end{center}
```

It is also possible to achieve this using a `minipage`. [11]

### 1.14.36 Verbatim In tabular

You cannot directly write `\verb{...}` or `\begin{verbatim}...\end{verbatim}` in a `tabular` environment. An easy solution is to use the following syntax:

```
\texttt{\textbackslash nameofacommand}
```

The `\texttt` command simply typesets its argument using the `\verb` font. That does the trick.

### 1.14.37 Verbatim In Footnotes

The same problem arises when you are writing verbatim text in footnotes. There are here two different approaches:

1. Use, as before, the command `\texttt`,
2. Use the `fancyvrb` package. After your preamble, just use `\VerbatimFootnotes`, the verbatim text being put between “+” symbols. For example, you may write

```
We can write verbatim\footnote{\verb+thistextisinverbformat+}
text in footnotes
```

once you have used `\VerbatimFootnotes`.

### 1.14.38 Verbatim In Hierarchy

Writing verbatim text in sections, subsections, subsubsections, ..., is also impossible. You may use `\texttt{}`.

**1.14.39 Vertically Aligned Itemize Environments**

Let us consider two pairs of vertically aligned itemize environments, from [11].

- |                           |                   |
|---------------------------|-------------------|
|                           | • first sentence  |
| • another first sentence  | • second sentence |
| • another second sentence | • third sentence  |
|                           |                   |
| • another first sentence  | • first sentence  |
| • another second sentence | • second sentence |
|                           | • third sentence  |

You can achieve this using a hbox such as in

```
\hbox{%
  \hbox to .52\linewidth{\vbox{\begin{itemize}
    \itemsep -.1em
    \item another first sentence
    \item another second sentence
  \end{itemize}}\hss}
  \hbox to .44\linewidth{\vbox{\begin{itemize}
    \itemsep -.1em
    \item first sentence
    \item second sentence
    \item third sentence
  \end{itemize}}\hss}}
```

## Fine Tuning

---

*Fine tuning* reveals its importance when writing documents. Two documents having the same content, but not the same appearance, won't be liked the same way by readers.

### Contents

---

<b>2.1</b>	<b>Avoiding</b>	<b>90</b>
2.1.1	Color In TOC And TOFs	90
<b>2.2</b>	<b>Counting</b>	<b>91</b>
2.2.1	Pages And Tables	91
	Example	92
2.2.2	Paragraphs	92
	Example	92
<b>2.3</b>	<b>Defining</b>	<b>92</b>
2.3.1	A Turning Page Text	92
2.3.2	Bold Index Numbers	93
2.3.3	Useful Commands	93
<b>2.4</b>	<b>Managing</b>	<b>94</b>
2.4.1	Beamer Presentations and Articles	94
2.4.2	Big Documents	96
2.4.3	Floats	97
2.4.4	Indentation	99
	With	99
	Without	99
2.4.5	Index	99
<b>2.5</b>	<b>Modifying</b>	<b>99</b>
2.5.1	QED Square	99
<b>2.6</b>	<b>Using</b>	<b>101</b>
2.6.1	Cleardoublepage Command Correctly	101
2.6.2	Crops	101
2.6.3	Microtype	101
2.6.4	Pedagogical Tools	102
	Gap Texts	102
2.6.5	Units From the SI	102

<b>2.7 Writing</b>	<b>103</b>
2.7.1 Comments In an Effective Way	103
2.7.2 Derivatives and Such Symbols In a Better Way	103
2.7.3 Letters With the <code>lettre</code> Documentclass	
Without a Big Space	104
2.7.4 Letters With the <code>lettre</code> Documentclass	
Without the Line Bending	104
2.7.5 MATLAB Code	105
2.7.6 Messages on Would-Be Blank Pages	106
2.7.7 Unhabitual Punctuation Marks	107
Interrobang	107
Percontation point (ironicon)	107
<b>2.8 Referring to \labels by long names</b>	<b>108</b>
2.8.1 Code	108
2.8.2 Example	109
<b>2.9 Personalizing The Index</b>	<b>109</b>
2.9.1 Standard Personalization	109
2.9.2 French Tricks	109
2.9.3 Insensitive Letter Sort	110
2.9.4 Special Letter Sort	110
2.9.5 German Letter Sort	110
2.9.6 Math Formulae Sort	111
2.9.7 Greek Letter Sort	112
2.9.8 Special Characters Sort	113
2.9.9 Last Refinements	113

---

## 2.1 Avoiding

### 2.1.1 Color In TOC And TOFs

With the `hypersetup` we declared at Section 1.14.19, p. 70, links (which are not necessarily URLs, think about internal links) will appear coloured in blue in the different tables: of contents, of figures, and of tables. It can be avoided easily by using such a code:

```

\begingroup
\hypersetup{linkcolor=black}
\tableofcontents
\endgroup

\newpage
\begingroup
\hypersetup{linkcolor=black}
\listoffigures
\endgroup

```

```

\newpage
\begingroup
\hypersetup{linkcolor=black}
\listoftables
\endgroup

```

You can evidently use the trivially-related option of the package.

## 2.2 Counting

### 2.2.1 Pages And Tables

It would be interesting to know the number of pages of a given document, simply by calling [\[93\]](#)

```
\ref{TotPages}
```

which would give as output the number of pages. In an analogous way, one would for example want to know the number of tables of the document. This can be achieved using

```
\AbsTables
```

But before using the latter command, we must declare (in the preamble)

```

\newcommand*{\OrigChapter}{}
\let\OrigChapter\chapter
\newcounter{abstables}
\renewcommand*{\chapter}{%
  \addtocounter{abstables}{\value{table}}%
  \OrigChapter%
}
\newcommand*{\AbsTables}{0}
\makeatletter
\AtBeginDocument{%
  \AtEndDocument{%
    \addtocounter{abstables}{\value{table}}%
    \immediate\write\@mainaux{%
      \string\gdef\string\AbsTables{\number\value{abstables}}%
    }%
  }%
}
\makeatother

```

For the former command, we need only

```
\usepackage{totpages}
```

in the preamble, for the pages.

We then use the two commands given above.

**Example**

There are 250 pages and 2 Table in this document.

**2.2.2 Paragraphs**

If you want to number your paragraphs, you can write

```
\newcounter{vcount}
\def\Header#1{\medskip
\hbox{\bfseries #1}
\setcounter{vcount}{1}
\everypar{\arabic{vcount}\stepcounter{vcount}\ }
}
```

in your preamble.

**Example**

Here is the output of this code, if one writes

```
\Header{The Title}
One line of text that is long enough to wrap as a paragraph that is long enough to wrap as a paragraph.

Two lines of text that are long enough to wrap as a paragraph that is long enough to wrap as a paragraph.

More lines of text that are long enough to wrap as a paragraph that is long enough to wrap as a paragraph.
```

**The Title**

1 One line of text that is long enough to wrap as a paragraph that is long enough to wrap as a paragraph.

2 Two lines of text that are long enough to wrap as a paragraph that is long enough to wrap as a paragraph.

3 More lines of text that are long enough to wrap as a paragraph that is long enough to wrap as a paragraph.

**2.3 Defining****2.3.1 A Turning Page Text**

On some examination sheets, some professors or lecturers like to indicate to the reader that the ‘page’ can be turned. That avoids oversights to the student, thereby avoiding them cold sweat at the end of the exam, realizing that there were other questions they needed to answer. (This happens more frequently than you might imagine!)

One might implement a L<sup>A</sup>T<sub>E</sub>X mechanism so that each odd page (except the last one, if it is odd) displays a message to turn the page. This mechanism can also be useful for different kinds of documents. This package can thus e.g. be used for exams, or special documents printed ‘twoside.’

To load the package, please use

```
\usepackage[options]{turnthepage}
```

The following options are available:

- **short**: will display ‘/...’ at the bottom of each odd page, in its right corner,
- **english**: will display ‘*Turn the page.*’ at the bottom of each odd page, in its right corner,
- **francais**: will display ‘*Tournez la page.*’ at the bottom of each odd page, in its right corner,
- **nederlands**: will display ‘*Sla de pagina om.*’ at the bottom of each odd page, in its right corner,
- **deutsch**: will display ‘*Bitte wenden.*’ at the bottom of each odd page, in its right corner.

As the package calls `\turnthepage` where it needs to be placed, you can define `\turnthepage` as you want. For example, if you want a more polite way to say it, you can use

```
\renewcommand{\turnthepage}{\itshape %
  Turn the page please.}
```

after the preamble. Thanks to Marc van Dongen [166] for his help about this package. For more info, please check [86].

### 2.3.2 Bold Index Numbers

In large books, especially physics’ books, some words are often used through all the book. Indexing every instance of the word thus produces an important number of pages for each word.

This is interesting, but there should normally be a place in the book where the word is explained, defined, or at least discussed in a more thoroughly fashion than elsewhere.

This place needs to be referenced too in the index, but it is important for the reader to distinguish it from simple citations.

A good technique which is generally used is to put in bold style this number. Assuming you are using the `makeidx` package, this can be achieved using

```
\index{TUGboat|textbf}
```

where, here, `TUGboat` is our index word, and where `textbf` evidently means that the related page number will be printed in bold in the index part of the book. Thanks to Peter Flynn for this trick [24].

### 2.3.3 Useful Commands

Other useful commands can be declared, such as

```

\newcommand{\U}{\cap}
\newcommand{\constant}{\mathrm{constant}}
\newcommand{\etc}{\textit{etc.}}
\newcommand{\ie}{\textit{i.e.}~}
\newcommand{\degr}{\mathrm{d\degree}}
\renewcommand{\deg}{\mathrm{\degree}}
\providecommand{\abs}[1]{\left|#1\right|}
\providecommand{\norm}[1]{\left\lVert#1\right\rVert}

```

The two last lines declared resp. one command `\abs` and one command `\norm`; both take only one argument, and return resp. the expressions of the modulus and the norm of their parameter.

For example, you can now write

$$\left|\sum_{i=-5}^n i\right|$$

simply using

```
\abs{\sum_{i=-5}^n i}
```

## 2.4 Managing

### 2.4.1 Beamer Presentations and Articles

When you use the `beamer` class, you can evidently produce slides (we will call this the presentation), e.g. for a conference, but you can also produce a handout.

Generally, a handout is considered to be a scaled version of the slides, so that more than one slide fits on the page. This can generally be achieved using PDF tools, and this is not our concern here.

The case we want to treat here is the one where you write a presentation, and want to write a report too, both being related to the same content. It might be interesting if the presentation you are writing mainly consists of a summary of your research.

If you want to reuse your beamer presentation to write the article, or the contrary, you can do this in a very simple way, thanks to Mr. Knudsen explanations at [54], and to [33]. Consider that you have three files:

- `content.tex`,
- `presentation.tex`,
- `report.tex`.

Then, the idea is to put the whole content (i.e. text that will either be shown in the presentation, or in the report, the ‘or’ being inclusive) in `content.tex`. That means that all your content will be put in this file. This file does not

need to have any preamble, because it will be included by both `report.tex` and `presentation.tex`.

Now, we need to specify how your content will be directed to the presentation, the report, or both. There is a quick way for this: `\mode<presentation>` for `presentation` mode, and `\mode<article>` for `article` mode. That means that you will have a file `content.tex` like

```
\section{First Section}
\mode<article> {
% Appears in the article
}
\frame {
\frametitle{My Frame}
% Appears in both versions
}
% Appears in both versions
% (except if specified differently)
```

You can then also use the `\mode` command in the `frame` content e.g. to scale differently images. Say you are in a `frame`. You can then use

```
\begin{figure}[h]
\centering
\mode<presentation> {
\includegraphics[scale=0.2]{img.eps} % small
}
\mode<article> {
\includegraphics[scale=1.5]{img.eps} % bigger
}
\caption{Appears in both versions.}
\end{figure}
```

The `report.tex` file will be written using e.g.

```
\documentclass{article}
\usepackage{beamerarticle}
\begin{document}
\input{content.tex}
\end{document}
```

The `presentation.tex` file will follow habitual rules:

```
\documentclass{beamer}
\begin{document}
\input{content.tex}
\end{document}
```

By using this scheme, your content will only reside in `content.tex` and it could be an advantage, for two main reasons:

- You have only one file for the content, for both versions,
- If you use many code snippets, in your report, from your presentation, content modification (e.g. modifying numbers, a caption, a title, ...) is centralized, and, as a result, you only modify things once. This allows extra-consistency between your report and your slides.

### 2.4.2 Big Documents

If you are writing a big document, it is easier for its maintenance to structure it internally, *i.e.* using different files for different purposes. (The coherent structure of a document is not in the scope of this document.) An example of a good structure for sections which are not too long would be given by:

1. `mainfile.tex`: your main file: it will have such a content:

```
% usepackage declarations

% other commands you want here

\begin{document}

% \input declarations
% or
% \include declarations

\end{document}
```

2. `chapter1/section1.tex`: the first section of the chapter 1,

3. `:`

4. `chaptern/sectionn.tex`: the  $n^{\text{th}}$  section of the  $n^{\text{th}}$  chapter.

End each `tex` document you `input` or `include` in `mainfile.tex` by `\endinput`; this tells  $\text{\LaTeX}$  that this is the end of the input.

You will use two commands to put (virtually) the content of a `sectionn.tex` file into `mainfile.tex`:

1. `\include{sectionn.tex}`: it basically does

```
\input{sectionn.tex}
\newpage
```

2. `\input{sectionn.tex}`: it basically puts the content of the file `sectionn.tex` in your `mainfile.tex`, like a `#include` directive in the C programming language.

### 2.4.3 Floats

Many  $\text{\LaTeX}$ -beginners are often frustrated. This happens especially when, after compilation, their **figures**, **tables**, and, more generally, floats, appear at incongruous places. For example, one might already have a ‘sufficient’ number of floats on one page, the following floats being moved to the following page(s).

On a content-oriented approach, it is, effectively, sometimes disturbing, be it to the reader, or to the writer, to look at such behaviors without any clue on how to interfere with this placement.

Before trying anything, an important thing is to understand, briefly, why and how  $\text{\LaTeX}$  manages to put floats this way. It is typographically disadvised to put too much images or tables on one page, or, more generally, in the ‘main matter’ of a document. If you need to put so much images in your document, ask yourself some questions:

- Is it a good way to present things?
- Shouldn’t I select the most important images and tables?
- Couldn’t I put some images and tables on specific pages only, or in the appendices?
- ...

Once you have clear and responsible answers to these questions, you might still need to fine tune the placement of your floats.

If you need to put floats on specific float pages, in the ‘main matter’ of your document, you might use the `[p]` option after the beginning of your environment: **figure**, **table**, etc.

However, typography rules sometimes go against your rules, i.e. the way you (want to) present things. If you know what you are doing, you might simply, sometimes, feel the need to include a float at a place where  $\text{\LaTeX}$  does not want it to be at all.

If you tried even with the `[h!]` option, meaning ‘put this float here!’, to no avail, you might consider modifying some internal parameters, because putting an exclamation mark in the list of placement options makes  $\text{\LaTeX}$  ignore all the constraints but

`\topfraction`, `\bottomfraction`, and the ones with `floatpage` in their names. [190]

Once again, ask you questions:

- Are there too much floats on my page?
- Is my float too big?
- Where am I including my float?

If your float is too big, either you resize it until it comes on the desired page, or you let it where  $\text{\LaTeX}$  puts it. But if there are too many floats, you need to know that  $\text{\LaTeX}$  defines a limit on the number of floats by page.

Acceptable parameter modifications might be found at [190]:

```

% See p.105 of "TeX Unbound" for suggested values.
% See pp. 199-200 of Lamport's "LaTeX" book for details.
% General parameters, for ALL pages:
\renewcommand{\topfraction}{0.9} % max fraction of floats at top
\renewcommand{\bottomfraction}{0.8} % max fraction of floats at bottom
% Parameters for TEXT pages:
\setcounter{topnumber}{2}
\setcounter{bottomnumber}{2}
\setcounter{totalnumber}{4}
\setcounter{dbltopnumber}{2} % for 2-column pages
\renewcommand{\dbltopfraction}{0.9} % fit big float above 2-col. text
\renewcommand{\textfraction}{0.07} % allow minimal text w. figs
% Parameters for FLOAT pages:
\renewcommand{\floatpagefraction}{0.7} % require fuller float pages
% N.B.: floatpagefraction MUST be less than topfraction!
\renewcommand{\dblfloatpagefraction}{0.7} % require fuller float pages
% remember to use [htp] or [htpb] for placement

```

If you find that liberal values of the float parameters still are causing trouble, you can try forcing a float page with `\clearpage` to disgorge the accumulated blockage. If you do not want to force a pagebreak, use the `afterpage` package and tell  $\text{\LaTeX}$  `\afterpage{\clearpage}`, which should force a float page when the current page comes to an end. If floats continue to pile up at the end, you probably have one too big to fit on a page; try reducing its size. [190] In extreme cases, you might consider using the `float` package. [190]

$\text{\LaTeX}$  mechanisms are sufficiently well done so that, under normal use, they do not cause you too much trouble with floats, i.e. floats are not rejected too far in the document. In my reports, where I often need to include graphics, I generally encounter no problem, without any specific tuning.

In a general use, the most frequently question that the user asks is

‘Do I need to let  $\text{\LaTeX}$  put my floats e.g. two pages after where they are included, between some paragraphs which have absolutely no link with these figures, or should I manage to put my figures differently?’

The answer to this question is that you should normally consider using pages for floats, or at least manage to put figures in pages whose context is linked to the appearing figures. If a float is rejected on the following page, it is generally not too much shocking for the reader, and it might even be a good thing if you prefer this way to present things. But if your floats are rejected further, it generally becomes disturbing for the reader, who needs to look for figures through your document.

### 2.4.4 Indentation

#### With

Being conscious of what you are doing, you know that you want an indentation for some reason. Just use

```
\indent
% your text
```

#### Without

Being conscious of what you are doing, you know that you want no indentation for some reason. Just use

```
\noindent
% your text
```

### 2.4.5 Index

In large books, especially physics' books, some words are often used through all the book. Indexing every instance of the word thus produces an important number of pages for each word. This is interesting, but there should normally be a place in the book where the word is explained, defined, or at least discussed in a more thoroughly fashion than elsewhere. This place needs to be referenced too in the index, but it is important for the reader to distinguish it from simple citations.

A good technique is to put in bold style this number. Assuming you are using the `makeidx` package, this can be achieved using

```
\index{TUGboat|textbf}
```

where, here, `TUGboat` is our index word, and where `textbf` evidently means that the related page number will be printed in bold in the index part of the book. Thanks to Peter Flynn for this trick [24].

## 2.5 Modifying

### 2.5.1 QED Square

One would for example want to modify the traditional  $\square$  into  $\blacksquare$ . It is achieved easily using

```
\usepackage{amssymb}
\renewcommand\qedsymbol{$\blacksquare$}
```

You can use the symbol of your choice, but most common are

■ <code>\blacksquare</code>	♥ <code>\heartsuite</code>	△ <code>\triangle</code>
♣ <code>\clubsuit</code>	♠ <code>\spadesuit</code>	
◇ <code>\Diamond</code>		

If you want the QED square to be put right (at the right), you can simply use

```
\def\qedsymbol {% set up

\widowpenalty=10000 % so we dont break the page before \square
\displaywidowpenalty=10000 % ditto
%
% horizontal
\leavevmode % \nobreak means lines not pages
\hfil % ragged right if we spill over
\hskip.2em % ensure some space
\hfill % push \square to right
$\blacksquare$ % the end-of-proof mark
%
% vertical
\par}} % build paragraph
```

in your preamble, for example for a `blacksquare` symbol. This comes from an adaptation of Mr. (Paul) Taylor's QED package whose manual is available at <http://www.paultaylor.eu/proofs/QEDdoc.pdf> the style file being at <http://www.paultaylor.eu/proofs/QED.sty>

It is evidently suggested to use this package at the place of such a manual code. You can also use a different solution:

```
\def\qedsymbol{%
  \mbox{}
  \nolinebreak
  \hfill
  $\diamond$ % the qed symbol
  \medbreak
  \par
}
```

where the `\medbreak` is not mandatory. You can then use ponctually `\qedsymbol`, or, better, use a package which does it for you (such as `ntheorem`).

## 2.6 Using

### 2.6.1 Cleardoublepage Command Correctly

According to [175], if you are using a two-side mode, you have the ability to get a new odd-side page by using `\cleardoublepage`. However,  $\text{\LaTeX}$  does not give you the ability to get a new even-side page. To do so, put

```
\usepackage{ifthen}
\usepackage{color}
\newcommand{\newevenside}{%
\ifthenelse{\isodd{\thepage}}{\newpage}{%
\newpage
\textcolor{white}{placeholder}%
\thispagestyle{empty}%
\newpage%
}%
}
```

in the preamble. To active the new even-side page, type `\newevenside` where you want it to be.

If the given page is an odd-side page, the next new page is subsequently an even-side page, and  $\text{\LaTeX}$  will do nothing more than a regular `\newpage`. However, if the given page is an even page,  $\text{\LaTeX}$  will make a new (odd) page, put in a placeholder, and make another new (even) page.

### 2.6.2 Crops

Displaying crops can be useful for typographic purposes. For a given document, just add

```
\usepackage{geometry}
\geometry{paperheight=210mm,paperwidth=200mm}
\usepackage[cam,a4,center,info,graphics]{crop}
\newcommand*\cropfont[1]{\small\textup{\textmd{\textsf{#1}}}}
\crop[font=cropfont]
```

to the preamble.

### 2.6.3 Microtype

According to [112], the `microtype` package provides a  $\text{\LaTeX}$  interface to the micro-typographic extensions of  $\text{\pdfTeX}$ : most prominently, character protrusion and font expansion, furthermore the adjustment of interword spacing and additional kerning, as well as hyphenatable letterspacing (tracking) and the possibility to disable all or selected ligatures.

That is, `microtype` basically helps you to produce a better output when used conjointly with `pdfTeX`.

You do not have to worry too much about it, just use

```
\usepackage[protrusion=true,draft=false,
final,verbose=true,babel=true]{microtype}
```

in the preamble of your document. If you do not use `babel`, you may write

```
babel=false
```

or delete this entry from the options passed to `microtype`.

### 2.6.4 Pedagogical Tools

#### Gap Texts

There are some (most often pedagogical, but it can happen with forms too, for example) documents where you would like to produce at least one version, with dots at the place of the text which has to be written by somebody else. If you are making a form, you do not necessarily need another version with the dots being completed with the “rights words,” as they simply do not exist: they depend on who the form is addressed to.

If you are making some very primitive test for your students, or something which needs these two versions, you can use my `dashundergaps` package (see [78]).

### 2.6.5 Units From the SI

The *SI* (“Système International”) defines the most used measure units. At the place of using constructs such as

```
2~\mathrm{kg}
```

to denote the mass of 2 kilograms, one should use the `siunitx` package in such a way:

```
\usepackage[allowzeroexp=true,obeymode=true,redefsymbols,textcelsius,
textdegree,textminute,textmu,load={abbr,addn},decimalsymbol=comma]{siunitx}
```

and use `\SI{106.5e2}{m}` (resp. `\si{m}`) to display  $106,5 \times 10^2$  m (resp. m). No \$ are needed. The advantages of such a method is that

1. It will keep a coherent displaying of these values, when you could have forgotten one ~ when writing using the old method,
2. It makes the tedious task of typesetting units and values automatic.

## 2.7 Writing

### 2.7.1 Comments In an Effective Way

You may have noticed in the examples of this book that when a bracket is opened, thus leading to `{`, a `%` character (that is, the comment character) is written. To take a very basic example, one could write in a math environment:

```
\sum_{%
i=0}^{%
n}i.
```

to show, for example, to the reader that the sum’s subscript begin at the following line, *i.e.* “check the next line: continued . . .” That is the same concept in

```
\vbox{%
\hskip2.8em$\overbrace{\hphantom{b \hspace{2\arraycolsep} \ldots
\hspace{2\arraycolsep} c}}^{n \mbox{\scriptsize\ times}}}$
% other commands
}
```

That is also the same in

```
\begingroup%
```

and

```
\begin@SpecialObj%
\pssetxlength\pst@dimg{1cm}
\pssetylength\pst@dimh{1cm}
```

There is also a L<sup>A</sup>T<sub>E</sub>X reason for this. Briefly, `%` starts a comment that goes to the end of the line. The normal effect is that it does not insert the space (or a `\par`) from the newline. [126]

### 2.7.2 Derivatives and Such Symbols In a Better Way

One often writes

$$\int_a^b f(x)dx,$$

but this is pretty ugly. Compare with

$\int_a^b f(x) \, dx$
-----------------------

There are two differences between the second and the first expression:

1. A space is put between  $f(x)$  and  $dx$ ,
2.  $dx$  is displayed at the place of  $dx$ .

The first point is important, as it enhances readability. The second is even more crucial, as it shows that the variable is not  $dx$  but  $x$ . Everybody knows it, but it is more beautiful.

The second one is written with

```
\int_{a}^{b}f(x)\,\mathrm{d}x
```

where `\,` denotes the space, and `\mathrm` only works for the character which follows it. If you want it to work for everything which follows it, use brackets: `\mathrm{thisisinromanfont}`.

### 2.7.3 Letters With the `lettre` Documentclass Without a Big Space

If you use the `lettre` class, you might dislike the space between address and the beginning of the text of the letter. To make this space a lot smaller, simply use

```
\setlength\openingspace{-1cm}
```

in the preamble. Thanks to Gonzalo Medina Arellano for this [76].

### 2.7.4 Letters With the `lettre` Documentclass Without the Line Bending

The `lettre` documentclass draws a small line bending to help you bending the letter in three parts once you have printed it. If you find this line bending disgracious, there are three methods to remove it [163]:

1. Put

```
\makeatletter
\newcommand*\NoRule{}\renewcommand*\rule@length{0}
\makeatother
```

in the preamble. You can then use `\NoRule` after the beginning of the `letter` environment;

2. In the `.ins` file, you can add a line which will be evaluated each time you use this `.ins` file:

```
\renewcommand*\rule@length{0}
```

3. You can also define a new class, `xletter.cls`, for example, defined like this:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{xlettre}

\newcommand*\xlettre@do{}
\newcommand*\xlettre@rule{}
\newcommand*\xlettre@norule{%
```

```

\let \xlettre@institut=\institut
\def \institut ##1{%
  \xlettre@institut{##1}%
  \def \rule@length {0}%
}%
\def \@institut {%
  \makeatletter \input{default.ins}\makeatother
  \def \rule@length {0}%
}%
}

\DeclareOption{rule}{\let \xlettre@do =\xlettre@rule}
\DeclareOption{norule}{\let \xlettre@do =\xlettre@norule}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{lettre}}

\ExecuteOptions{norule}
\ProcessOptions

\LoadClass{lettre}

\xlettre@do

\endinput

```

You can then use either

```
\documentclass[norule]{xlettre}
```

or

```
\documentclass{xlettre}
```

or

```
\documentclass[rule]{xlettre}
```

### 2.7.5 MATLAB Code

To write *MATLAB code* efficiently [53, 67], the best idea is to use the `listings` package together with<sup>1</sup> `mcode`. Thus, you may put

```

\usepackage{listings}
\usepackage[bw,numbered,framed,final]{mcode}

```

in the preamble of your document. The following options are available for `mcode`:

---

<sup>1</sup> Note that this package may be downloaded at <http://files.myopera.com/locksley90/blog/mcode.sty>, at [http://web.mit.edu/~paul\\_s/www/14.170/matlab/mcode.sty](http://web.mit.edu/~paul_s/www/14.170/matlab/mcode.sty) or even at [53].

- `bw` is useful if you intend to print the document,
- `numbered` is useful if you want line numbers to be written before each line of code,
- `framed` is useful if you want a frame around the source code blocks,
- `final` is useful if you have “globally” set the `draft` option, as `listings` will not, in such a case, output the code at all. That forces it to do so anyway.

You are then able to put a MATLAB source code using

```
\lstinputlisting{/path_to_mfile/yourmfile.m}
```

or placing snippets of source code in a `lstlisting` environment. For example, you would then do

```
\begin{lstlisting}
% Example of Matlab code for calculating hypotenuse
% § $c=\sqrt{a^2+b^2}$ §
a = 3;
b = 4;
c = sqrt(a^2+b^2);
\end{lstlisting}
```

Note that “§” allow you to “escape” from L<sup>A</sup>T<sub>E</sub>X mode. As a result, you are not obliged anymore to pass lots of parameters to `listings` using `lstset`.

This will give a better presentation than using `lstlisting` together with a declaration like

```
\lstset{language=MATLAB,basicstyle=\small\ttfamily,showstringspaces=false,%
numbers=left,commentstyle=\itshape,backgroundcolor=\color{white},%
stepnumber=2,numbersep=5pt,escapeinside={(*@}{@*)}}
```

### 2.7.6 Messages on Would-Be Blank Pages

On would-be blank pages, one should ask for a message such as “This page intentionally left blank.” For this, just use

```
\makeatletter
\def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
\vspace*{\fill}
\hfill
\begin{center}
This page intentionally left blank.
\end{center}
\vspace{\fill}
\thispagestyle{empty}
\newpage
\if@twocolumn\hbox{}\newpage\fi\fi\fi}
\makeatother
```

in the preamble. The pages which would otherwise be left blank will now contain this message.

### 2.7.7 Unhabitual Punctuation Marks

One might feel the need for unhabitual (yet sometimes useful) punctuation marks such as the interrobang and the percontation point (ironicon).

I here adapt the point of view of the English (in the large sense) writer; Spanish-oriented well-known characters will not be exposed here, because already mastered by the public.

Note that I found none of the two further described symbols in [103].

#### Interrobang

This symbol is generally used when a question is formulated in an excited manner, expresses excitement or disbelief in the form of a question, or asks a rhetorical question. [180] An example of situation where this would be of interest is given at [180]:

‘How much did you pay for those shoes?’

The Unicode code point is U+203D. The interrobang can be displayed in L<sup>A</sup>T<sub>E</sub>X by using the package `textcomp` and the command `\textinterrobang` [180]:

⸔

#### Percontation point (ironicon)

Though in the English language there is no standard accepted method to denote irony or sarcasm in written conversation, several forms of punctuation have been proposed.

Among the oldest and frequently attested are the percontation point (*punctus percontativus*, also known as an ironicon) invented by Henry Denham in the 1580’s, and the irony mark, furthered by Alcanter de Brahm in the 19th century. [181]

Both of these marks were represented visually by a backwards question mark (unicode U+2E2E reversed question mark). [181]

Using the `graphicx` package, `\reflectbox?` works pretty well [21]:

⸕

One might define a command such as `\ironicon` by e.g. `\newcommand{\ironicon}{\reflectbox?}`. That being done, you can now directly use `\ironicon` in your document.

## 2.8 Referring to \labels by long names

### 2.8.1 Code

Say you want to `\label{}` e.g. an `enumerate`'s environment's item. You then have, for example,

```
\begin{enumerate}
\item First item,
\item Second item.
\end{enumerate}
```

Now, one wants to say 'see Item 1.' This can be achieved using

```
\begin{enumerate}
\item First item, \label{item:firstitem}
\item Second item.
\end{enumerate}
```

and then, 'see Item [Fruits and Co.](#)' somewhere in the text. This is the traditional approach to the `\ref` and `\label` commands. But now, how does one manage if he wants to write 'see Item `\ref{item:firstitem}`' and wants 'see Item [name]' to appear, where [name] is an attribute of `\label`, for example defined like

```
\begin{enumerate}
\item First item, \mylabel{item:firstitem}{%
Fruits and Co.}
\item Second item.
\end{enumerate}
```

where here, name is 'Fruits and Co.'? He could then have, in the output, 'see Item Fruits and Co.'

For this, I did not try using `zref`, but I used Mr. Oberdiek's trick [\[71\]](#):

```
\makeatletter
\newcommand*{\mylabel}[2]{%
  \@bsphack
  \begingroup
    \def\@currentlabel{#2}%
    \label{#1}%
  \endgroup
  \@esphack
}
\makeatother
```

### 2.8.2 Example

This gives

1. First item,
2. Second item

and a reference to the first item is ‘[Fruits and Co..](#)’

## 2.9 Personalizing The Index

### 2.9.1 Standard Personalization

When generating an index with `makeindex`, one could create a `perso.ist` file, and put

```
heading_prefix "{\\bfseries\\hfil "
heading_suffix "\\hfil}\\nopagebreak\\n"
headings_flag 1
delim_0 "\\dotfill"
delim_1 "\\dotfill"
delim_2 "\\dotfill"
```

into it to write the first alphabet symbol in bold font, and use dots as delimiters. This is the index’s personalizing file. It is generally used jointly with `makeindex` using

```
makeindex -s perso.ist filename.idx
```

where `filename.idx` has been created when executing `latex` on `filename.tex`.

### 2.9.2 French Tricks

If your document is in French, you could ask for “Symboles” at the place of “Symbols” and “Nombres” at the place of “Numbers.” This is achieved by appending

```
symhead_positive "Symboles"
symhead_negative "symboles"
numhead_positive "Nombres"
numhead_negative "nombres"
```

to the previous code.

### 2.9.3 Insensitive Letter Sort

If you want, for example, an insensitive letter sort, you may use, according to [37]:

```
sort_rule "A" "a"
sort_rule "B" "b"
sort_rule "C" "c"
sort_rule "D" "d"
sort_rule "E" "e"
sort_rule "F" "f"
sort_rule "G" "g"
sort_rule "H" "h"
sort_rule "I" "i"
sort_rule "J" "j"
sort_rule "K" "k"
sort_rule "L" "l"
sort_rule "M" "m"
sort_rule "N" "n"
sort_rule "O" "o"
sort_rule "P" "p"
sort_rule "Q" "q"
sort_rule "R" "r"
sort_rule "S" "s"
sort_rule "T" "t"
sort_rule "U" "u"
sort_rule "V" "v"
sort_rule "W" "w"
sort_rule "X" "x"
sort_rule "Y" "y"
sort_rule "Z" "z"
```

### 2.9.4 Special Letter Sort

For T<sub>E</sub>X-style umlaut-macros, you may use, according to [36]:

```
sort_rule "\\\"A" "ae"
sort_rule "\\\"a" "ae"
sort_rule "\\\"O" "oe"
sort_rule "\\\"o" "oe"
sort_rule "\\\"U" "ue"
sort_rule "\\\"u" "ue"
sort_rule "\\ss({})?" "ss"
```

### 2.9.5 German Letter Sort

For German L<sup>A</sup>T<sub>E</sub>X-style umlaut-macros, you may use, according to [36]:

```

sort_rule "\"A" "ae"
sort_rule "\"a" "ae"
sort_rule "\"O" "oe"
sort_rule "\"o" "oe"
sort_rule "\"U" "ue"
sort_rule "\"u" "ue"
sort_rule "\"s" "ss"

```

### 2.9.6 Math Formulae Sort

If you use things like `\index{log@\texttt{log}}`, and other fancy things, you may use, according to [39]:

```

% first remove enclosing '$'-characters
*merge_rule "\$(.*)\$" "\1"

% function-name macros will be sorted like the function they stay for
merge_rule "\\log" "log"
merge_rule "\\lg" "lg"
merge_rule "\\ln" "ln"
merge_rule "\\lim" "lim"
merge_rule "\\limsup" "limsup"
merge_rule "\\liminf" "liminf"
merge_rule "\\sin" "sin"
merge_rule "\\arcsin" "arcsin"
merge_rule "\\sinh" "sinh"
merge_rule "\\cos" "cos"
merge_rule "\\arccos" "arccos"
merge_rule "\\cosh" "cosh"
merge_rule "\\tan" "tan"
merge_rule "\\arctan" "arctan"
merge_rule "\\tanh" "tanh"
merge_rule "\\cot" "cot"
merge_rule "\\coth" "coth"
merge_rule "\\sec" "sec"
merge_rule "\\csc" "csc"
merge_rule "\\max" "max"
merge_rule "\\min" "min"
merge_rule "\\sup" "sup"
merge_rule "\\inf" "inf"
merge_rule "\\arg" "arg"
merge_rule "\\ker" "ker"
merge_rule "\\dim" "dim"
merge_rule "\\hom" "hom"
merge_rule "\\det" "det"

```

```

merge_rule "\\exp" "exp"
merge_rule "\\Pr" "Pr"
merge_rule "\\gcd" "gcd"
merge_rule "\\deg" "deg"

```

### 2.9.7 Greek Letter Sort

For Greek letters, you may use, according to [39]:

```

% the pronunciation of Greek letters decides their sort order
merge_rule "\\alpha" "alpha"
merge_rule "\\beta" "beta"
merge_rule "\\gamma" "gamma"
merge_rule "\\delta" "delta"
merge_rule "\\epsilon" "epsilon"
merge_rule "\\zeta" "zeta"
merge_rule "\\eta" "eta"
merge_rule "\\theta" "theta"
merge_rule "\\iota" "iota"
merge_rule "\\kappa" "kappa"
merge_rule "\\lambda" "lambda"
merge_rule "\\mu" "mu"
merge_rule "\\nu" "nu"
merge_rule "\\xi" "xi"
merge_rule "\\pi" "pi"
merge_rule "\\rho" "rho"
merge_rule "\\sigma" "sigma"
merge_rule "\\tau" "tau"
merge_rule "\\upsilon" "upsilon"
merge_rule "\\phi" "phi"
merge_rule "\\chi" "chi"
merge_rule "\\psi" "psi"
merge_rule "\\omega" "omega"
merge_rule "\\varepsilon" "epsilon"
merge_rule "\\vartheta" "theta"
merge_rule "\\varpi" "pi"
merge_rule "\\varrho" "rho"
merge_rule "\\varsigma" "sigma"
merge_rule "\\varphi" "phi"
merge_rule "\\Gamma" "Gamma"
merge_rule "\\Delta" "Delta"
merge_rule "\\Theta" "Theta"
merge_rule "\\Lambda" "Lambda"
merge_rule "\\Xi" "Xi"
merge_rule "\\Pi" "Pi"

```

```

merge_rule "\\Sigma" "Sigma"
merge_rule "\\Upsilon" "Upsilon"
merge_rule "\\Phi" "Phi"
merge_rule "\\Psi" "Psi"
merge_rule "\\Omega" "Omega"
merge_rule "\\aleph" "aleph"

```

### 2.9.8 Special Characters Sort

According to [40], you may use

```

% special characters come first
sort_rule "\." "\b\."
sort_rule "\:" "\b\: "
sort_rule "\", " "\b\, "
sort_rule "\;" "\b\; "
sort_rule "\!" "\b\! "
sort_rule "\?" "\b\? "
sort_rule "\' " "\b\ ' "
sort_rule "\" " "\b\ \" "
sort_rule "\+ " "\b\+ "
sort_rule "\- " "\b\ - "
sort_rule "\% " "\b\% "
sort_rule "&" "\b\& "
sort_rule "/" "\b\ / "
sort_rule "(" "\b\ ( "
sort_rule ")" "\b\ ) "
sort_rule "@" "\b\ @ "
sort_rule " " "\b\ "

```

to handle special characters correctly.

### 2.9.9 Last Refinements

If the commands `\LaTeX` and `\TeX` are not correctly handled by your `makeindex`, you may use, according to [38, 41]:

```

merge_rule "\\LaTeX" "LaTeX"
merge_rule "\\TeX" "TeX"

```

This page intentionally left blank.

## Various Tricks

### 3.1 Automatically Capitalizing First Letters of Words

Many editors require you to capitalize the first letter of each (‘nontrivial’) word that appears in a section/subsection, etc. This process might be cumbersome, tedious, and might result in inconsistencies: you might consider one time a word as ‘trivial,’ and another time not.

Several implementations were proposed, and you might have a look at [\[120\]](#) and [\[133\]](#).

### 3.2 Changing “References” to Some Other Text

At [\[137\]](#), Anand asked how to change the “References” title to an arbitrary name. For example, you might have (as he did) only one reference in your document. Generally, either using

```
\renewcommand\bibname{Reference}
```

or

```
\renewcommand\refname{Reference}
```

will do the trick, as explained at [\[137\]](#). For example, `\bibname` is used in the `book` documentclass.

### 3.3 Coloring the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> Logo

I saw at [\[140\]](#) a method to color the following logos: T<sub>E</sub>X, L<sup>A</sup>T<sub>E</sub>X, L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>. It makes extensive use of the `xcolor` package, and was posted by Mr. Gonzalo Medina.

### 3.3.1 Code

```

\usepackage{xcolor}

\makeatletter
\newcommand*\ClrTeX[3]{%
  \textcolor{#1}{T}\kern-.1667em%
  \lower.5ex\hbox{%
    \textcolor{#2}{E}}%
    \kern-.125em\textcolor{#3}{X}\@}

\newcommand*\ClrLaTeX[5]{%
  \textcolor{#1}{L}\kern-.36em%
  {\sbox\z@ T%
    \vbox to\ht\z@{\hbox{\check@mathfonts
      \fontsize\sf@size\z@
      \math@fontsfalse\selectfont
      \textcolor{#2}{A}}}%
    \vss}%
  }%
  \kern-.15em%
  \ClrTeX{#3}{#4}{#5}}

\newcommand*\ClrLaTeXe[7]{%
  \mbox{\m@th
    \if b\expandafter\@car%
      \f@series\@nil\boldmath\fi
      \ClrLaTeX{#1}{#2}{#3}{#4}{#5}\kern.15em%
      \textcolor{#6}{2}$_{\textstyle%
        \textcolor{#7}{\varepsilon}}$}}
  \makeatother

```

### 3.3.2 Example

One can then use e.g.

```
\ClrTeX{red}{green}{blue}
```

to get (a colored) **TeX**,

```
\ClrLaTeX{red}{green}{blue}{gray}{magenta}
```

to get (a colored) **LaTeX**, and

```
\ClrLaTeXe{red}{green}{blue}{gray}{magenta}
{olive}{purple}
```

to get (a colored) **LaTeX<sub>2 $\epsilon$</sub>** .

### 3.4 Correct Spacing Around Square Brackets In Math Mode

Let us consider some math outputs, together with their code equivalents, at Table 3.1. These come from [11].

No.	Output	Command
1.	$c = ]a, b[ = d$	<code>c = ]a,b[ = d</code>
2.	$c = ]a, b[ = d$	<code>c = {\]a,b\} = d</code>
3.	$c = ]a, b[ = d$	<code>c = \mathopen]a,b\mathclose[ = d</code>
4.	$c = [a, b] = d$	<code>c = [a,b] = d</code>
5.	$c   d$	<code>c   d</code>
6.	$c \mid d$	<code>c \mathrel{\mid} d</code>
7.	$c \mid d$	<code>c \mid d</code>
8.	$x \in ]-\infty, 0[$	<code>x \in ]-\infty, 0[</code>
9.	$x \in ]-\infty, 0[$	<code>x \in {\]}\infty, 0[</code>
10.	$x \in ]-\infty, 0[$	<code>x \in {\]}\infty, 0[</code>

**Table 3.1.** Some math outputs.

Some of these outputs look great, when others do not:

1. is ugly, because the equality sign is too close to the brackets,
2. , 3., 4. are correct, because the equality sign is correctly placed. Approach 2. seems to be the most obvious way to such an output,
5. is generally considered as ugly (but this might be useful in some specific situations),
6. , 7. are correct, because the  $|$  is sufficiently far from both letters ( $c$  and  $d$ ),
8. is ugly, because the  $\in$  sign is too close to the set,
9. is ugly too,
10. is correct.

### 3.5 Hiding the Column Separator in multicol

It might come in handy to hide the column separator in `multicol` environment's output. As Mr. Torbjorn explained it at [136], the width of the line is decided by the `columnseprule` parameter. As a result, using

```
\setlength{\columnseprule}{0pt}
```

will remove the line.

### 3.6 How Do You Choose Between Declaring a New Command Or a New Environment?

Let us consider the problem of deciding if you need to declare a new command or a new environment in  $\text{\LaTeX}$  when trying to automatize some task.

With some experience, the problem to choose between both options is easily solved. However, it might appear difficult for the beginner. For example, a ‘Question’ block like a ‘Theorem’ block needs to be an environment. But why?

First, it would look weird to typeset a whole mathematical example in a command. Second, you have probably never seen such an implementation. Third, there is Mr. Wright’s answer at [149].

‘For a command `\foo{<stuff>}`,  $\text{\TeX}$  reads all of `<stuff>` before expanding `\foo`. This can be an issue if `<stuff>` is very big. Also, once  $\text{\TeX}$  has read an argument, it can be difficult or impossible to change the category codes [...]. In particular, `verbatim` material is difficult to handle reliably in an arbitrary argument.

With environments,  $\text{\LaTeX}$  does not read everything in one go. At `\begin{foo}` the set up is done, but the environment is then typeset normally without everything being read in one go. This makes it possible to include `verbatim` and avoids filling up  $\text{\TeX}$ ’s memory with too much.  $\text{\LaTeX}$  makes each environment a group, which is why things like

```
\begin{center}
  \bfseries
  Text
\end{center}
```

keep the font changes local.’

As Mr. Goutet points out, what can be done with a command can also be issued with an environment, but the converse is false.

### 3.7 Naming Different Rows or Columns Of a Matrix Using Braces

After having explored

- how to write an adjacency matrix in [80],
- how to write a brace over one row of elements in [83],

here is how to ‘label’ (not in the  $\text{\LaTeX}$  sense: labelling here means naming) different extern elements of a matrix.

### 3.7.1 Code

Different solutions were discussed at [124] but I found that the less hackish one was jfbu's one. Here it is. First include the `amsmath` and the `mathtools` packages, then define four possibilities of braces:

```
\newcommand\coolover[2]{\mathrlap{\smash{%
\overbrace{\phantom{\begin{matrix} #2 %
\end{matrix}}}\sim\mbox{${\#1$}}}}\#2}

\newcommand\coolunder[2]{\mathrlap{\smash{%
\underbrace{\phantom{\begin{matrix} #2 %
\end{matrix}}}\_ \mbox{${\#1$}}}}\#2}

\newcommand\coolleftbrace[2]{%
#1\left\{\vphantom{\begin{matrix}%
#2 \end{matrix}}\right.}

\newcommand\coolrightbrace[2]{%
\left.\vphantom{\begin{matrix}%
#1 \end{matrix}}\right\}\#2}
```

You can now e.g. use

```
\[ \vphantom{
\begin{matrix}
\overbrace{XYZ}\sim\mbox{${R$}}}%
\\ \\ \\ \\ \\ \\
\underbrace{pqr}_\mbox{${S$}}
\end{matrix}}%
\begin{matrix}
\vphantom{a}\\
\coolleftbrace{A}{e \\ y\\ y}\\
\coolleftbrace{B}{y \\ i \\ m}
\end{matrix}%
\begin{bmatrix}
a & \coolover{R}{b & c & d} \%
& x & \coolover{Z}{x & x}\\
e & f & g & h & x & x & x \\
y & y & y & y & y & y & y \\
y & y & y & y & y & y & y \\
y & y & y & y & y & y & y \\
i & j & k & l & x & x & x \\
m & \coolunder{S}{n & o} \%
& \coolunder{W}{p & x & x} & x
\end{bmatrix}%
\begin{matrix}%
```

```

\coolrightbrace{x \ x \ y \ y}{T}
\coolrightbrace{y \ y \ x \ x}{U}
\end{matrix}

```

### 3.7.2 Example

This gives the following output:

$$\begin{array}{c}
 \left. \begin{array}{c} A \\ B \end{array} \right\} \left[ \begin{array}{cccccc}
 \overset{R}{a} & \overset{R}{b} & \overset{R}{c} & \overset{Z}{d} & \overset{Z}{x} & \overset{Z}{x} \\
 e & f & g & h & x & x \\
 y & y & y & y & y & y \\
 y & y & y & y & y & y \\
 y & y & y & y & y & y \\
 i & j & k & l & x & x \\
 \underset{S}{m} & \underset{W}{n} & \underset{W}{o} & \underset{W}{p} & \underset{W}{x} & \underset{W}{x}
 \end{array} \right\} \begin{array}{c} T \\ U \end{array} .
 \end{array}$$

## 3.8 Overfull \hboxes

You might have already seen those black rectangles in some  $\text{\LaTeX}$  outputs. These black rectangles basically allow you to know if ‘text comes partially in the margin.’ (That is the simple explanation, though.)

To see these rectangles, simply set `\overfullrule` to e.g. 1 mm [121]:

```
\overfullrule=1mm
```

## 3.9 Using the III Symbol for Maths

The III symbol (Sha) is a letter of the Cyrillic alphabet. It is widely used in Mathematics; some examples include [183]

- *Algebraic Geometry*: the Tate-Shafarevich group of an Abelian variety  $A$  over a field  $K$  is denoted  $\text{III}(A/K)$ , a notation first suggested by J. W. S. Cassels. (Previously it had been unimaginatively denoted TS.)
- *Shuffle Algebra*: the shuffle product is often denoted by III. Consider two words  $X$  and  $Y$ ; the sum of all the ways of interlacing them is  $X \text{III} Y$ . [184]
- *Signal processing*: the III ‘function’ is what is commonly referred to as a Dirac comb.

I will here treat about one of its applications: Signal Processing. You can see the III symbol, in its upper and lower case forms at Fig. 3.1.



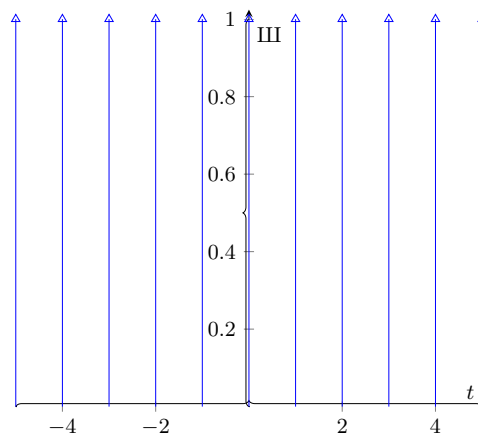
**Fig. 3.1.** Upper and lower case forms of the III symbol.

### 3.9.1 III in Signal Processing

In Signal Processing, as we introduced before,  $\text{III}(t)$  is the Dirac Comb ‘function.’ That is, if we define  $\text{III}(t) = \text{III}_1(t)$  as the 1-periodic Sha function, we also have

$$\text{III}_T(t) = \sum_{k \in \mathbb{Z}} \delta(t - kT),$$

which leads to a serie of Dirac impulses where each impulse is separated by  $T$  units from the two surrounding other impulses, as illustrated by Fig. 3.2.



**Fig. 3.2.** The  $\text{III}(t)$  ‘function.’

### 3.9.2 III in L<sup>A</sup>T<sub>E</sub>X

#### Making it Appear

We consider that the reader simply wants to use this character in his documents, for math purposes. As the III symbol is also in UTF-8, you can type it directly in your document, provided your document is, for example, encoded in UTF-8. Same remarks apply for its lower case counterpart iii. Apart from the `input encoding`, there is also the `font encoding` which might cause you troubles. Here, you have several choices, but the best is to use the `T2A` option for the `fontenc` package. Provided you have a good installation, you should get Type 1 fonts for the III symbol in your PDF using it. As a result, use

```
\usepackage[utf8]{inputenc}
\usepackage[T2A]{fontenc}
```

in your document's preamble if you want to typeset the III symbol.

#### Using It in Math Mode

Litteraly enclosing the III symbol between `$` and `$` or simply putting it in a math environment, e.g. in an `equation` will result to an absence of any III symbol because there is no corresponding symbol for math in the font encoding used. As a result, different solutions are possible, but the simplest is to declare it as a math operator, and to put the character in `\mbox`:

```
\DeclareMathOperator{\sha}{\mbox{III}}
```

Other alternatives are possible. For example, you might use

```
\DeclareSymbolFont{cyrletters}
{OT1}{wncyr}{m}{n}
\DeclareMathSymbol{\sha}
{\mathalpha}{cyrletters}{"no}
```

in the preamble, where `no` is the corresponding number from the OT1 to output III. [\[168\]](#)

#### Using It in BibT<sub>E</sub>X

As you might still be using BibT<sub>E</sub>X, you might already wonder how to output a bibliography entry with the III symbol. This is achieved by using a compatible `.bib` file encoding (here, I used UTF-8, but it could have been a different one, such as `koi8-r`).

### 3.10 ‘Why is L<sup>A</sup>T<sub>E</sub>X Separated Into Many Thousands of Packages?’

I came across [\[150\]](#), where an assistant Linux admin (namely, `jphenow`) asked why L<sup>A</sup>T<sub>E</sub>X is separated into so many packages. Summarizing the answers, L<sup>A</sup>T<sub>E</sub>X

- i. should be seen as composed of many ‘applications.’ These are the packages.
- ii. was initially created to run on low-memory machines, in the eighties.
- iii. ’s packages are provided by users and not by the LaTeX core team, and, as a result, many packages have been created, because there are many users, and many ideas.
- iv. ’s packages need to be understood just like Perl modules.

This page intentionally left blank.

## A Special Case Study: The Springer svmono Class

We describe here the problems one often faces when using the old `svmono` class from Springer. New editions should not cause any troubles.

### Contents

---

<a href="#">4.1 Introduction</a> .....	125
<a href="#">4.2 The Problem Of Tocdepth</a> .....	126

---

#### 4.1 Introduction

This book is written using the `svmono` documentclass. It thus leads to

```
\documentclass{svmono}
```

in the preamble. This should be noted, as finding the class and the instructions for this documentclass were difficult.

The `svmono.cls` file is the class file of the `svmono` class. It has some built-in parameters' declarations. For example,

```
\setlength\oddsidemargin {63\p@}
\setlength\evensidemargin {63\p@}
\setlength\marginparwidth {90\p@}
\setlength\headsep        {12\p@}

\setlength{\parindent}    {15\p@}
\setlength{\parskip}      {\z@ \@plus \p@}
\setlength{\hfuzz}        {2\p@}
\setlength{\arraycolsep}  {1.5\p@}
```

has a clear meaning.

## 4.2 The Problem Of Tocdepth

Despite rendering beautiful documents, the Springer Verlag's old editions of `svmono` class do not respect `tocdepth`. As a result, subparagraphs cannot be included in the Table Of Content, *i.e.* TOC (which includes only chapters, sections, subsections, subsubsections, and paragraphs but not subparagraphs). The following code solves this issue:

```
\makeatletter
\renewcommand\subparagraph{%
  \@startsection{subparagraph}{4}{\z@}%
  {-18\p@}% \p@lus -4\p@ \@minus -4\p@}%
  {6\p@}% \p@lus 4\p@ \@minus 4\p@}%
  {\normalfont\normalsize\itshape\righskip=\z@ \@plus 8em\pretolerance=10000}%
}
\makeatother
```

Once you have put this in the preamble of your document, the `tocdepth` problem is solved.

## Part II

---

### Special Documents



L<sup>A</sup>T<sub>E</sub>X can be used for other purposes than “just” advanced typesetting: you may want to write envelopes, posters, ...

Being able to write simple or advanced documents of one kind is a good thing, but there are only rare events in life where you need to do *one thing* in particular, such as making reports and reports. For example, if you make a paper for a conference, using a poster could be a good way to illustrate your comments. But you have no idea at all about how to write such a document, and you do not have as much time as to look on the Internet for various tricks that you are not even sure to be correct.

Writing such documents is, for a L<sup>A</sup>T<sub>E</sub>X *user* (not for a L<sup>A</sup>T<sub>E</sub>X real programmer), the achievement of the best of his knowledge. With such documents, you can express your ideas in a freer way, because you are not as much limited as you were with, for example, a typical thesis `documentclass`.

These tricks are different from the tricks which were exposed in Part I, from p. 5 as you are here making *fine tuning*: you are not satisfied with the output you worked so hard for. Principally, these tricks will allow you to be prouder about your documents and to make what you want with L<sup>A</sup>T<sub>E</sub>X. The L<sup>A</sup>T<sub>E</sub>X language *has* to adapt to your document, and not the opposite.

This page intentionally left blank.

## AutoMath

*AutoMath* is a name I have decided to assign to macros which allow you to typeset exercises for students: generating equations, square roots of numbers, systems of linear equations, and so on. For every exercise's generation, one needs to add

```
\usepackage{lcg,calc,ifthen}
```

in the preamble, which means that three packages are loaded: `lcg`, `calc` and `ifthen`. The first one aims at generating pseudorandom numbers with a simple linear congruential generator, the second implements basic Arithmetic, and the third allows easier constructs for evaluation of expressions. Generating systems of equations and square roots, is slightly inspired from [\[15\]](#).

*All the solutions which will be automatically generated are not the most elegant ones, but render the arithmetical operations which were needed to compute the solution, thus leading to a better (in the pedagogical sense) answer.*

### 5.1 Generating Equations

#### 5.1.1 Linear

##### Example

To generate equations where the polynomial has a degree which equals 1, one can use the following code:

```
\reinitrand[first=-20, last=20, counter=a] \rand % two integers in ]-20, 20[ are generated
\newcommand{\randomlieq}{%
\chgrand[counter=b] \rand
% Replace 0 by another number
\whiledo{\value{a}=0}{\chgrand[counter=a] \rand}
\whiledo{\value{b}=0}{\chgrand[counter=b] \rand}

Solve the following equation:
\[
```

```

\thea x +\theb = 0.
\]
}

\randomlieq

```

It results in:

Solve the following equation:

$$-7x + 15 = 0.$$

Doing a solver for such a mathematical problem would be uninteresting, as it is really trivial to do.

### 5.1.2 Square

#### Example

To generate equations where the polynomial has a degree which equals 2, one can use the following code:

```

\newcounter{delta}

\reinitrand[first=-20, last=20, counter=a] \rand % three integers in )-20, 20( are generated
\newcommand{\randomsseq}{%
\chgrand[counter=b] \rand
\chgrand[counter=c] \rand
% Replace 0 by another number
\whiledo{\value{a}=0}{\chgrand[counter=a] \rand}
\whiledo{\value{b}=0}{\chgrand[counter=b] \rand}
\whiledo{\value{c}=0}{\chgrand[counter=c] \rand}

Solve the following equation:
\[
\thea x^2 +\theb x + \thec = 0.
\]
\solverandomsseq
}

\newcommand{\solverandomsseq}{%

\setcounter{delta}{\value{b}*\value{b}-4*\value{a}*\value{c}}
Solution:
\ifthenelse{\value{delta} < 0}{[x=\frac{-\arabic{b}\pm i\sqrt{-\arabic{delta}}}{2\times \arabic{a}}]\]}{\ifthenelse{\value{delta} > 0}
{\[x=\frac{-\arabic{b}\pm\sqrt{\arabic{delta}}}{2\times \arabic{a}}\]}\}}
}

\randomsseq

```

It results in:

Solve the following equation:

$$2x^2 + 7x + -16 = 0.$$

Solution:

$$x = \frac{-7 \pm \sqrt{177}}{2 \times 2}$$

## 5.2 Generating Square Roots

When one needs to typeset Mathematics exercises, a very basic exercise is to generate square roots that the student has to simplify manually.

### 5.2.1 Example

Using

```
\newcounter{expr} % each final expression
\chgrand[counter=B] \rand
\newcommand{\randomrac}{%
\reinitrand[first=2, last=20, counter=A] \rand
\setcounter{expr}{\value{B}*\value{A}*\value{A}}
$\sqrt{\theexpr}$}

\begin{tabular}{rl}
\hline
Square root to simplify & Student's solution\\
\hline
\randomrac & \dotfill\\
\randomrac & \dotfill\\
\randomrac & \dotfill\\
\hline
\end{tabular}
```

results in:

Square root to simplify Student's solution	
$\sqrt{1156}$	.....
$\sqrt{324}$	.....
$\sqrt{1600}$	.....

## 5.3 Generating Systems (Of Linear Equations)

Linear systems are common exercises, even at the beginning of the undergraduate studies. They are tedious to solve, especially when you did a lot during your studies.

### 5.3.1 Two Equations, Two Variables

We here deal with linear systems of two equations and two variables. They can be easily generated and solved (if the principal determinant does not equal 0) using Cramer's rule, with the following code:

```

\newcounter{dettwo} % principal determinant
\newcounter{detcramertwotimestwox} % determinant for x
\newcounter{detcramertwotimestwoy} % determinant for y

\reinitrand[first=-20, last=20, counter=a] \rand % six integers in )-20, 20( are generated
\newcommand{\randmtwotimestwoSYS}{%
\chgrand[counter=b] \rand
\chgrand[counter=c] \rand
\chgrand[counter=d] \rand
\chgrand[counter=e] \rand
\chgrand[counter=f] \rand

% Replace 0 by another number
\whiledo{\value{a}=0}{\chgrand[counter=a] \rand}
\whiledo{\value{b}=0}{\chgrand[counter=b] \rand}
\whiledo{\value{d}=0}{\chgrand[counter=d] \rand}
\whiledo{\value{e}=0}{\chgrand[counter=e] \rand}

% The sentences are different according to det
\setcounter{dettwo}{\value{a}*\value{e}-\value{b}*\value{d}}
\ifthenelse{\value{dettwo}=0}{%
{Explain why does the following system has not a unique solution:}%
{Solve the following system:}
\[[
\begin{cases}
\thea x+\theb y \&= \thec \\\
\thed x+\thee y \&= \thef \\\
\end{cases}
\]
\solvettwotimestwoSYSTEM
}

\newcommand{\solvettwotimestwoSYSTEM}{%
\setcounter{detcramertwotimestwox}{\value{c}*\value{e}-\value{b}*\value{f}}
\setcounter{detcramertwotimestwoy}{\value{a}*\value{f}-\value{c}*\value{d}}

\ifthenelse{\NOT{\value{detcramertwotimestwox}=0} \AND{\NOT{\value{detcramertwotimestwoy}=0}}}{Solution:
\begin{align*}
x&=\frac{\begin{vmatrix} \arabic{c} & \arabic{b} \\ \arabic{f} & \arabic{e} \end{vmatrix}}{\begin{vmatrix} \arabic{a} & \arabic{b} \\ \arabic{d} & \arabic{e} \end{vmatrix}} \\
y&=\frac{\begin{vmatrix} \arabic{a} & \arabic{c} \\ \arabic{d} & \arabic{f} \end{vmatrix}}{\begin{vmatrix} \arabic{a} & \arabic{b} \\ \arabic{d} & \arabic{e} \end{vmatrix}} \\
\end{align*}
}{Cramer cannot work here.}
}

\randmtwotimestwoSYS

```

Note that there are lots of different ways to solve linear systems, but the simplest implementation of a basic solver has to use Cramer's rule for simplicity's sake. It then results in:

Solve the following system:

$$\begin{cases} 2x + -17y = 9 \\ 9x + 7y = -18 \end{cases}$$

Solution:

$$x = \frac{\begin{vmatrix} 9 & -17 \\ -18 & 7 \end{vmatrix}}{\begin{vmatrix} 2 & -17 \\ 9 & 7 \end{vmatrix}} = \frac{-243}{167}$$

$$y = \frac{\begin{vmatrix} 2 & 9 \\ 9 & -18 \end{vmatrix}}{\begin{vmatrix} 2 & -17 \\ 9 & 7 \end{vmatrix}} = \frac{-117}{167}$$

If the solution has to be hidden, just comment the

`\solvetwotimestwosystem`

by putting a % just at the beginning of this line.

### 5.3.2 Three Equations, Three Variables

The same code can be adapted for three times three systems. Here is the code without solver:

```
\newcounter{detthree} % principal determinant
\newcounter{detcramerthreetimesthreeex} % determinant for x
\newcounter{detcramerthreetimesthreey} % determinant for y
\newcounter{detcramerthreetimesthreez} % determinant for z

\reinitrand[first=-50, last=50, counter=a] \rand % twelve integers in )-50, 50( are generated
\newcommand{\randomthreetimesthreesys}{%
\chgrand[counter=b] \rand
\chgrand[counter=c] \rand
\chgrand[counter=d] \rand
\chgrand[counter=e] \rand
\chgrand[counter=f] \rand
\chgrand[counter=g] \rand
\chgrand[counter=h] \rand
\chgrand[counter=i] \rand
\chgrand[counter=j] \rand
\chgrand[counter=k] \rand
\chgrand[counter=l] \rand

% Replace 0 by another number
\whiledo{\value{a}=0}{\chgrand[counter=a] \rand}
\whiledo{\value{b}=0}{\chgrand[counter=b] \rand}
\whiledo{\value{c}=0}{\chgrand[counter=c] \rand}
\whiledo{\value{e}=0}{\chgrand[counter=e] \rand}
\whiledo{\value{f}=0}{\chgrand[counter=f] \rand}
\whiledo{\value{g}=0}{\chgrand[counter=g] \rand}
```

```

\whiledo{\value{i}=0}{\chgrand[counter=i] \rand}
\whiledo{\value{j}=0}{\chgrand[counter=j] \rand}
\whiledo{\value{k}=0}{\chgrand[counter=k] \rand}

% The sentences are different according to det
\setcounter{detthree}{\value{a}*\value{f}*\value{h}-\value{a}*\value{g}*\value{j}}
-\value{b}*\value{e}*\value{h}+\value{b}*\value{g}*\value{i}+\value{c}*\value{e}*\value{j}-\value{c}*\value{f}*\value{i}}
\ifthenelse{\value{detthree}=0}{%
{Explain why does the following system has not a unique solution:}%
{Solve the following system:}
\begin{cases}
\thea x+\theb y +\thec z &=& \thed \\
\thee x+\thef y +\theg z &=& \theh \\
\thel x+\thej y +\thek z &=& \thel
\end{cases}
\end{cases}
}

\randomthreetimesthreesys

```

It results in this:

<p>Solve the following system:</p> $\begin{cases} 1x + -13y + 33z &= -50 \\ -1x + -44y + 7z &= -11 \\ 50x + 20y + 5z &= -3 \end{cases}$
---

Note that Yves Delhayé (see <http://yvesdelhayé.be/>), <http://www.pyromaths.org/> and <http://chingatome.net> allow you to use exercises in L<sup>A</sup>T<sub>E</sub>X too.

## Envelopes

You are decided to write envelopes with L<sup>A</sup>T<sub>E</sub>X but do not know how. Here is one global solution. What could be better looking than a L<sup>A</sup>T<sub>E</sub>X-generated envelope?

### Contents

---

<b>6.1</b>	<b>C6 Standard And Adaptations</b>	<b>137</b>
6.1.1	In the Printer	138
6.1.2	Output	138
<b>6.2</b>	<b>Standards</b>	<b>139</b>

---

### 6.1 C6 Standard And Adaptations

Here is an example of an envelope code (C6 standard):

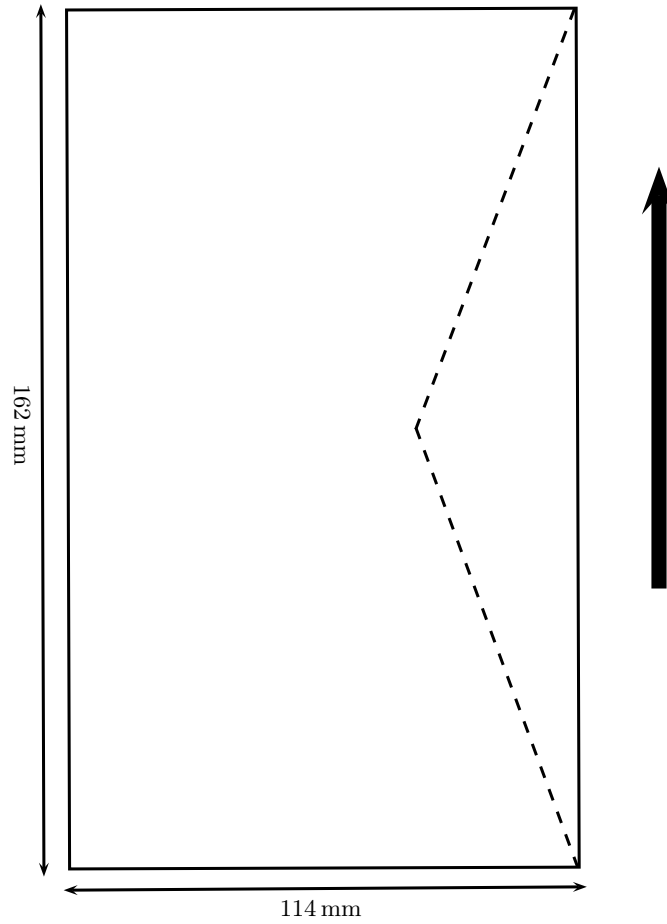
```
\documentclass[12pt]{letter}
\usepackage{geometry}
\geometry{paperheight=162mm,paperwidth=114mm}
\usepackage{graphics}
\usepackage[c6envelope,noprintbarcodes,richtenvelopes,printreturnaddress]{envlab}
\makelabels
\begin{document}
\startlabels
\mlabel{%
  FirstName LastName\\Street No\\ZIPcode City (Country)}{ % Sender's info
  FirstName LastName\\Street No\\ZIPcode City (Country)} % Receiver's info
\end{document}
```

You can directly print the output of such a code on a C6 envelope. If your envelopes match a height of  $M$  mm and have a width of  $N$  mm, just replace the third line in this code into:

```
\geometry{paperheight=Mmm,paperwidth=Nmm}
```

### 6.1.1 In the Printer

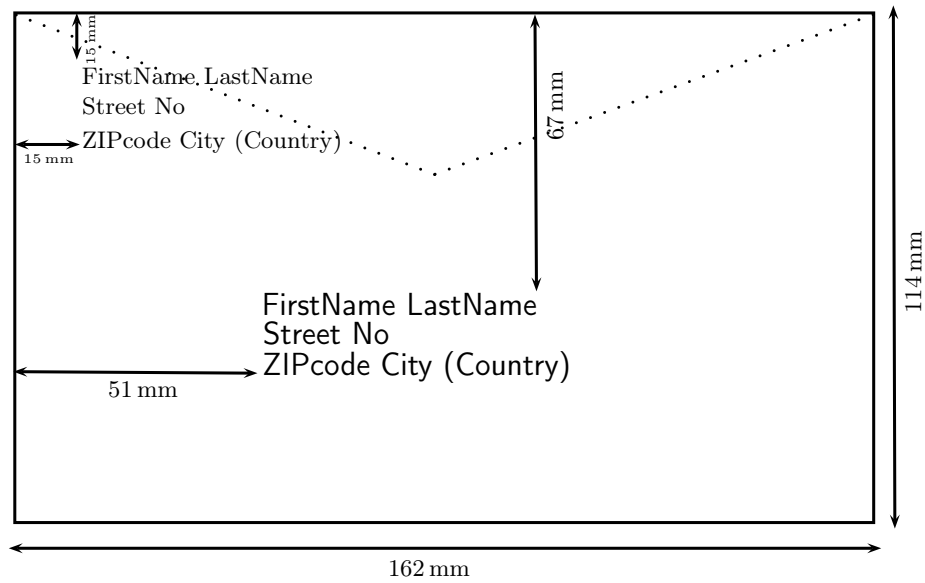
You need to put your C6 envelope in your printer as illustrated at Figure 6.1.



**Fig. 6.1.** How to put your C6 envelope in the printer.

### 6.1.2 Output

An example of the envelope's output which will be produced is given at Figure 6.2.



**Fig. 6.2.** C6 generated envelope.

## 6.2 Standards

Here are some examples of envelopes standards:

1. C4 format: 22,9 cm  $\times$  32,4 cm, to transport an A4 sheet,
2. C5 format: 16,2 cm  $\times$  22,9 cm, to transport an A4 sheet plied one time,
3. C6 format: 11,4 cm  $\times$  16,2 cm, to transport an A4 sheet plied two times,

You can directly modify the envelope of Section 6.1 to match the dimensions of your envelope.

This page intentionally left blank.

## Letters

Letters are still an important way of communicating, but they are less and less used nowadays. The most of the letters which are sent nowadays are coming from disadvantaged people who do not have a computer, or are sent for official purposes. We give here the most traditional  $\text{\LaTeX}$  codes which aim at helping users to typeset letters in an easy and standard way. Furthermore, you wouldn't send a not  $\text{\LaTeX}$ -typeset letter in a  $\text{\LaTeX}$ -typeset envelope, would you?

### Contents

---

<b>7.1</b>	<b>English</b>	<b>141</b>
<b>7.2</b>	<b>French</b>	<b>142</b>

---

### 7.1 English

The most traditional way to write an English letter with  $\text{\LaTeX}$  is to inspire from this code:

```
\documentclass[10pt,twoside]{letter}
\usepackage[english]{babel}

\begin{document}
\begin{letter}{~}

\name{FirstName}

\address{FirstName LastName\\
        Nb., Street\\
        C-ZIP City (Country)
    }
```

```

\telephone{123456789}

\vspace{-5cm}

\opening{Hello Mr.,} % or Ms.

% message's body

\closing{Sincerely yours,}

\end{letter}

\end{document}

```

## 7.2 French

The most traditional way to write a French letter with  $\text{\LaTeX}$  is to inspire from this code:

```

\documentclass[10pt,twoside]{lettre}
\usepackage[frenchb]{babel}

\begin{document}
\begin{letter}{~} % you can turn the ‘~’ into a message.

\name{FirstName LastName}

\address{FirstName LastName\\
         Nb., Street\\
         C-ZIP City (Country)
        }
\lieu{Place}
\telephone{0123456789} % if you want to give your phone number
%\notelephone % if you do not want to give any phone number
\fax{987654321} % if you want to give your fax number
%\nofax % if no fax

\vspace{-5cm}

\opening{Bonjour Monsieur,} % or ‘Madame’

% message's body

```

```
\closing{Respectueusement,}
```

```
\end{letter}
```

```
\end{document}
```

This page intentionally left blank.

## Posters

There are various ways to write posters with L<sup>A</sup>T<sub>E</sub>X. They are sometimes not straightforward solutions, but are always well documented on various Internet websites. Here are some ways to do this.

### Contents

<b>8.1 With Some Color</b> .....	<b>145</b>
8.1.1 Using Beamer .....	145
8.1.2 Notes on Other Approaches .....	147

## 8.1 With Some Color

### 8.1.1 Using Beamer

Take a look at the JACOBS University’s “Template for L<sup>A</sup>T<sub>E</sub>X poster using the Computational Physics and Biophysics Group Style,” *i.e.* [164]. Assuming you have the necessary files (which can be downloaded on [164]) to make it compile, use the following minimal code:

```
%=====
%== template for LATEX poster =====
%=====
%
%
%--A0 beamer slide-----
\documentclass[final]{beamer} % use beamer
\usepackage[orientation=portrait,
            size=a0,           % poster size
            scale=1.2          % font scale factor
            ]{beamerposter}    % beamer in poster size
%
%--some needed packages-----
```

```

\usepackage[american]{babel} % language
\usepackage[utf8]{inputenc} % standard linux encoding
%
%==The poster style=====
\usetheme{cpbgposter} % JACOBS' poster style
%--set colors for blocks (without frame)-----
\setbeamercolor{block title}{fg=ngreen,bg=white}
\setbeamercolor{block body}{fg=black,bg=white}
%--set colors for alerted blocks (with frame)-----
%--textcolor = fg, backgroundcolor = bg, dblue is the jacobs blue
\setbeamercolor{block alerted title}{fg=white,bg=dblue!70}%frame color
\setbeamercolor{block alerted body}{fg=black,bg=dblue!10}%body color
%
%==Titel, date and authors of the poster=====
\title{First part of the title\\
      Second part}
\author{FirstName LastName}
\institute[Your University]
        {Its Address1,
         Address2}
\date{\today}
%
%==some useful commands=====
% |x>
\newcommand{\ket}[1]{\left\vert\right.\right\rangle}
% <x|
\newcommand{\bra}[1]{\left\langle\right.\left\rangle}
% <x|y>
\newcommand{\braket}[2]{\left< #1 \vphantom{#2}\right< #1 \vphantom{#2} \right>}
% <x|a|y>
\newcommand{\sandwich}[3]{\left< #1 \vphantom{#2 #3} \right|
                        #2 \left|\vphantom{#1 #2} #3 \right>}
% d/dt
\newcommand{\ddt}{\frac{d}{dt}}
% D/Dx
\newcommand{\pdd}[1]{\frac{\partial}{\partial #1}}
% |x|
\newcommand{\abs}[1]{\left|\right.}
% k_x
\newcommand{\kv}[1]{\mathbf{k}_{#1}}
%=====
%==the poster content=====
%=====
\begin{document}

```

```

%--the poster is one beamer frame, so we have to start with:
\begin{frame}[t]
%--to divide the poster in columns we can use the columns environment
\begin{columns}[t] % the [t] options aligns the columns content at the top
%--the left column-----
\begin{column}{0.28\paperwidth}% the right size for a 3-column layout
%--abstract block-----
\begin{alertblock}{Abstract}
  Here is the abstract.
\end{alertblock}
\vskip2ex
\end{column}
%==big rightcolumn=====
\begin{column}{0.60\paperwidth} % the big right column
%--Examples block-----
\begin{block}{Examples}
  \ldots
\end{block}
\vskip2ex
%==two right columns=====
\begin{columns}[t,totalwidth=0.60\paperwidth]
\begin{column}{0.28\paperwidth}
\end{column}
\begin{column}{0.28\paperwidth}
\end{column}
\end{columns}
\end{column}
\end{columns}
\end{frame}
\end{document}

```

It can be easily modified to make posters looking a little bit different. An example of a poster created using Beamer is the poster of my [\[77\]](#) paper.

### 8.1.2 Notes on Other Approaches

You can make posters using really tricky code, but you really need to know what you are doing. If there is a problem with it, you are alone during a given time. Thus, it is always better to use easy approaches such as the Beamer one, to build posters.

This page intentionally left blank.

## Watermarking

For different purposes, it can be necessary to watermark your documents. For example, if you want to redistribute them but be the only person to possess the “original” (*i.e.* not watermarked) documents, this is a solution.

### Contents

---

<b>9.1</b>	<b>The Current Document</b>	<b>149</b>
9.1.1	The <code>background</code> Package	149
9.1.2	The <code>xwatermark</code> Option	150
9.1.3	The <code>TikZ</code> Way	150
9.1.4	The <code>draftcopy</code> package	150
<b>9.2</b>	<b>An Included PDF Document</b>	<b>151</b>

---

## 9.1 The Current Document

There are basically four different ways to put watermarks in your  $\text{\LaTeX 2}\epsilon$  documents:

1. The `background` package,
2. The `xwatermark` package,
3. The `TikZ` package,
4. The `draftcopy` package.

We shall discuss these different options separately now. We assume that the user wants to watermark all the pages. (Modifications so that only some subset of the set of pages is watermarked are easy.)

### 9.1.1 The `background` Package

A first solution to use watermarks (namely, the `background` package) has been developed by Mr. Gonzalo Medina Arellano. This package basically uses `Tikz`. The user can control many aspects (contents, position, color, opacity) of the background material that will be displayed. [75]

### 9.1.2 The xwatermark Option

To watermark the *current* tex document, you can simply put [94]

```
\usepackage[printwatermark=true,allpages=true,fontfamily=pag,
color=gray,grayness=0.9,mark=Draft,angle=45,
fontsize=5cm,markwidth=\paperwidth,
fontseries=b,scale=0.8,xcoord=0,ycoord=0]{xwatermark}
```

in your preamble, where parameters are modified in the obvious way.

### 9.1.3 The TikZ Way

You can also use TikZ [154]:

```
\begin{tikzpicture}[remember picture,overlay]
\node[rotate=0,scale=15,text opacity=0.1]
  at (current page.center) {Draft};
\end{tikzpicture}
```

writes the ‘Draft’ message in the center of the page, and

```
\begin{tikzpicture}[remember picture,overlay]
\node [xshift=1cm,yshift=1cm] at (current page.south west)
[text width=7cm,fill=red!20,rounded corners,above right]
{
  This is a draft!
};
\end{tikzpicture}
```

puts ‘This is a draft!’ in a box, at the desired place. Both might be put outside of the preamble. In both cases, you evidently need to load the `tikz` package. There are many other options (please check the package’s manual).

The advantage of this option is that you can punctually call TikZ after or before some text to watermark the relative page, without knowing its number.

Another solution to use watermarks has been developed by Mr. Gonzalo Medina Arellano, which is the `background` package. This package basically uses Tikz. The user can control many aspects (contents, position, color, opacity) of the background material that will be displayed.

### 9.1.4 The draftcopy package

You can use the `draftcopy` package too. You can then specify the intensity of the gray, the range of pages for which the word ‘DRAFT’ is printed and where it is printed (across the page or at the bottom). The package’s feature are best described in its manual [169], but, roughly, using [169]

```
\usepackage[english, all, portrait,draft]{draftcopy}
```

should suit your needs.

## 9.2 An Included PDF Document

If you want to watermark a `towatermark.pdf` document using  $\text{\LaTeX}$ , you firstly need to create a `main.tex` document. You then write in `main.tex`:

```
\documentclass[12pt]{article}
\usepackage{pdfpages}
\usepackage[printwatermark=true,allpages=true,fontfamily=pag,
color=gray,grayness=0.9,mark=MYWATERMARK,angle=45,fontsize=5cm,
markwidth=\paperwidth,fontseries=b,scale=0.8,
xcoord=0,ycoord=0]{xwatermark}

\begin{document}

\includepdf[pages=1-thelastpageyouwant]{towatermark.pdf}

\end{document}
```

This page intentionally left blank.

## Encoding Problems

If you are collaborating with other persons using different OSes, or simply migrating from one platform to another, you may have troubles with accents and special characters, especially if the language is French, Polish, or another language which makes an extensive use of special characters.

Some persons sometimes mix up the words which are related to *encoding*. We shall give here a brief summary of how you need to deal with encoding and L<sup>A</sup>T<sub>E</sub>X. Thanks for both Robin Fairbairns', Philipp Lehman's, Günter Milde's, Philipp Stephani's and Dominik Waßenhoven's contributions at [19].

### 10.1 The `inputenc` Package: the Encoding of the Document

You might have heard that putting

```
\usepackage[encoding]{inputenc}
```

in the preamble of every document you write should avoid encoding problems, assuming `encoding` is the encoding of the related `.tex` file. This is partially true, as it will solve *most* encoding problems, but not all.

#### 10.1.1 Description

According to [113], the `inputenc` package maps certain characters to their corresponding T<sub>E</sub>X macros according to the encoding option you select.

#### 10.1.2 Encoding Choice

Consider the `encoding` parameter. If you choose to use `utf8x`, it will use the extended UTF-8 character set, and you will be able to type, among other languages, Greek. Although there are popularity differences, text encoding

is nowadays unrelated to the operating system since all modern operating system components (windowing systems, font systems, drawing engines, ...) are Unicode-based or at least Unicode-capable. [19] For the rest of this text, remember that UTF-8 is an encoding for Unicode.

The `utf8` encoding covers the code ranges for which there is a defined  $\text{\LaTeX}$  encoding, while `utf8x` covers code ranges for which there is a usable font. [19] You might then think that `utf8x` is the best solution, but it has many disadvantages.

Amongst them, a first problem is that the UTF-8 decoder of `utf8x` is more intrusive than the one of `utf8`. `utf8` has an expandable scanner, when the one in `utf8x` is non-expandable. There is more potential for conflicts with other packages in the latter case. [19]

A second problem is that `utf8x` uses the `ucs` package, which is no longer maintained, and which breaks important packages such as `csquotes`. [19]

### 10.1.3 Solutions?

You are definitely not lost, because there are also `inputenx` (a drop-in replacement of `inputenc`) and `luainputenc` (for 8-bit encoding on  $\text{\LaTeX}$ ). For comparison,  $\text{\XeTeX}$  and  $\text{\LuaTeX}$  are natively in UTF-8 mode (and never require `inputenc`). [19] If you use  $\text{\pdfTeX}$  and want basic UTF-8 support (e.g., for accented Latin characters), load `utf8enc.def`.

But, generally, `utf8` lets you, for example, type Arabic script, Cyrillic script, Czech, French, German, (not Greek, use `utf8x` for example), Italian, Polish, Spanish, etc.

### 10.1.4 Platforms

On Microsoft Windows ©, users tend to use either ISO-8859-1 (which is commonly referred to as `Latin-1`), or CP1252. The former is generally intended for “Western European” languages. In this case, you need to replace `encoding` by `latin1`. The latter is an eight-bit character encoding designed to cover languages that use the Cyrillic alphabet such as Russian, Bulgarian, Serbian Cyrillic and other languages (French, ...), which do not use the Cyrillic alphabet. It is the most widely used for encoding the Bulgarian, Serbian and Macedonian languages, for example.

The character set CP1252 uses some of those positions for printable characters. Thus, the CP1252 character set is not identical with ISO-8859-1, because CP1252 contains more symbols than ISO-8859-1. [56] This is why using `latin1` as an option of `inputenc` under Microsoft Windows © pose no problem.

If you want to stick with `inputenc`, you might also use `ansinew` for `inputenc`. Sticking with `latin1` should not pose any problem until you type

in alphabets that `latin1` does not understand, such as Cyrillic. In ISO-8859-1, code positions 128 – 159 are explicitly reserved for control purposes; they “correspond to bit combinations that do not represent graphic characters.”

That being said, as Unicode is state of the art, it is disadvised to continue using either CP1252 or ISO-8859-1. It is suggested to use Unicode whenever possible. Moreover, since UTF-8 is supported in a better way by the `inputenc` package (and thus by pdfTeX-based LaTeX documents) than other encodings, it is the only choice that can be recommended. As a preliminary conclusion, you would then better use `utf8` as `encoding` value for `inputenc`.

If you are switching to `utf8` because you are sharing L<sup>A</sup>T<sub>E</sub>X sources with others and want to avoid problems with `Latin1/15` vs. `CP1252` vs. `MacRoman` (or similar for Eastern European encodings), using `inputenc` with `utf8` will work fine. This is, once again, another reason to use `utf8`.

If you need “real” Unicode support (e.g., when mixing scripts), you would better use a Unicode-savvy engine. More specifically, Xe<sub>T</sub>E<sub>X</sub> and Lua<sub>T</sub>E<sub>X</sub> are the best options if you want UTF-8. As a conclusion, if you stick with normal L<sup>A</sup>T<sub>E</sub>X use, simply invoke `utf8` unless you really need `utf8x`, whatever your platform. [19, 56]

## 10.2 The Proper Encoding of the Document File

To avoid clashes, the best thing is to keep your document in the same encoding as the encoding `encoding`, which is the option of `inputenc`. This is what we mentioned in the previous section. But, now, we will investigate the different ways to type texts in L<sup>A</sup>T<sub>E</sub>X.

### 10.2.1 Directly Writing Characters Without Commands

Directly writing characters without their associated commands (for example “é” written with a `e` and an acute accent) poses no problem until there is no encoding clash. If `encoding` for `inputenc` matches the document encoding, and that the current architecture understands this encoding, there will thus be no problem. [19]

But if you need to share sources without modifying the encoding, using commands at the place of direct keystrokes is better. But keep in mind that it makes the code more unreadable, and that this is also very unnatural to type such characters like this. [19]

Consider for example a text written under Microsoft Windows ©. Reading it on a Linux UTF-8 workstation, and saving it as UTF-8, will result in many encoding clashes, with exotic symbols.

There are actually four kinds of persons:

1. *Those who stick with commands.* Commands will always be valid, and, if deprecated one day, using `renewcommand` or other structures will make no problem. Less readable, but more portable, [19]

2. *Those who use commands only when necessary.* These are persons who try to see which character from their keyboard is directly rendered, which one is not, and, for those which are rendered, they typeset them directly, and, for those which are not rendered (as now), they use commands. This is sometimes tedious, difficult, error-prone, but it appears natural to assume that a character that is keyed in generates appropriate output unless it belongs to a special category (like \ or %): documents are easy to typeset, read, and edit for users with a similar keyboard, [19]
3. *Those who use various tricks to make L<sup>A</sup>T<sub>E</sub>X behaves as they want, even if they want things that are contrary to the state-of-the-art.* This might not be the best solution, but it depends on what the tricks are, [19]
4. *Those who use Unicode whenever possible.* While sometimes difficult to input (a good text editor will help), this results in readable sources that will work across OS boundaries. [19]

The best thing which can be recommended is evidently to stick with commands, such as demonstrated before. There are lots of sources to find symbols. A well-known one is [102]. One sometimes needs to include packages for other symbols, though.

### 10.2.2 Converting a File to the Good Encoding

If, say, you are dealing with documents in another encoding, the best thing is to use the following procedure, assuming you are working with Linux (or with such a virtual machine):

1. Find their current encoding,
2. Know what their future encoding will be (generally, you would better choose UTF-8 for aforementioned reasons),
3. Be sure that the encodings are compatible (i.e. the target character set is a superset of the source character set). If this is not the case, you will lose information,
4. Execute, a sample file being `fileinoldencoding.tex`, and the same file, in its new encoding being `fileinnewencoding.tex`:

```
iconv -f oldencoding fileinoldencoding.tex -o fileinnewencoding.tex
```

where `oldencoding` could be, for example, `windows-1252`. You might make this process automatic, e.g. by creating a shell file (here it is `bash`):

```
#!/bin/bash
for i in *.tex
do
  iconv -f windows-1252 -t utf-8 -- "$i" > "$i.utf8" && mv -- "$i.utf8" "$i"
done
```

and executing this file in a folder containing `.tex` files. You can evidently modify this script or the aforementioned commands as you want, to use another encoding. By default, the encoding is `utf8` if this is your current locale, but if you want encoding `newencoding`, use

`-t newencoding`

5. Open the file in an editor, the editor being set to open files in the output encoding,
6. If you see strange characters, there is a problem, and check the procedure. If everything seems normal, you can modify the file, save the modifications, but everything under the new encoding,
7. Compile the file(s) with the good `inputenc` declaration, as explained above.

Note that other tools perform such operations, such as `recode`. [19]

## 10.3 Dealing With BiB Files

*Everything from this section comes from Philipp Lehman, Philipp Stephani, either from [19] or from [62].*

If you are using BiB<sub>T</sub>E<sub>X</sub>, you might also have problems. The first problem is that BiB<sub>T</sub>E<sub>X</sub> *cannot* handle UTF-8 and non-fixed-width encodings in general. This is probably one of the most common misunderstanding when it comes to Bib<sub>T</sub>E<sub>X</sub>. There is no way around this restriction. Bib<sub>T</sub>E<sub>X</sub> cannot deal with multi-byte encodings.

You might have already heard about `bibtex8`, a drop-in BiB<sub>T</sub>E<sub>X</sub> replacement which supports 8-bit input. While it cannot handle UTF-8 either, it can sort 8-bit input in a way that is actually useful in languages other than English, when supplied with a suitable `csf` file.

Once again, you will need to use ASCII notation. Traditional BiB<sub>T</sub>E<sub>X</sub> can only alphabetize ASCII characters correctly. If the bibliographic data includes non-ASCII characters, they have to be given in ASCII notation. For example, instead of typing a letter like ‘ä’ directly, you need to input it as `\"a`, using an accent command and the ASCII letter. This ASCII notation needs to be wrapped in a pair of curly braces. Traditional Bib<sub>T</sub>E<sub>X</sub> will then ignore the accent and use the ASCII letter for sorting.

Apart from it being inconvenient, there are two major issues with this convention. One subtle problem is that the extra set of braces suppresses the kerning on both sides of all non-ASCII letters. But, also, simply ignoring all accents may not be the correct way to handle them, for alphabetical reasons which depend on the language.

These are the major reasons why switching to `bibtex8`, the 8-bit implementation of Bib<sub>T</sub>E<sub>X</sub>, is advisable. It can sort in a case-sensitive way and it can handle (single byte) non-ASCII characters properly, provided that you supply it with a suitable `csf` file.

The `biblatex` package is also capable of handling conflicting encodings in `.tex` and `.bib` files, provided that you specify the encoding of the `.bib` file with the `bibencoding` package option. [62] For more details about some

sample configurations you might try, please have a look at [62]. A good advice is to use `biblatex` and `biber`. [19]

One might then think about converting his `.bib` files to, say, `latin1`, and include them using

```
\begingroup
\inputencoding{latin1}
\bibliography{nameofthebibfile}
\endgroup
```

This will evidently work, but this is disadvised. If you are under Microsoft Windows ©, you will not need to use these four lines, simply because you have great chances to deal with `latin1` files. But, under Linux (UTF-8), your `.bib` files will be defaulted to `utf8`, which means that these lines would do the trick.

We reviewed how to correctly encode L<sup>A</sup>T<sub>E</sub>X documents. We did neither speak about font encoding, nor about mapping multiple encodings into one. We assumed by ‘encoding’ the name ‘input encoding.’ We shall now take a quick look at these two subjects. The reader might also distinct these concepts from the language rules that need to be loaded appropriately for each language, using e.g.

```
\usepackage[language]{babel}
```

## 10.4 Font Encodings

Roughly, *font encoding* defines at which position inside a T<sub>E</sub>X-font each letter is stored.

The default L<sup>A</sup>T<sub>E</sub>X font encoding is `OT1`, the encoding of the original Computer Modern T<sub>E</sub>X font. It contains only the 128 characters of the 7-bit ASCII character set. [43] When accented characters are required, T<sub>E</sub>X creates them by combining a normal character with an accent. While the resulting output looks perfect, this approach stops the automatic hyphenation from working inside words containing accented characters. Besides, some of Latin letters could not be created by combining a normal character with an accent, to say nothing about letters of non-Latin alphabets, such as Greek or Cyrillic. To overcome these shortcomings, several 8-bit CM-like font sets were created. [176]

For more details, please check Chap. 16 and [176].

## 10.5 Mapping Multiple Encodings Into One

Multiple input encodings could be mapped into one font encoding, which reduces number of required font sets. Font encodings are handled through the `fontenc` package:

```
\usepackage[encoding]{fontenc}
```

where `encoding` is the font encoding. It is possible to load several encodings simultaneously. [176]

## 10.6 Summary

The beginner might be impressed by the complexity of things. However, this is not that complicated for european and american languages. For these languages, you will generally use `utf8` as an `encoding` for `inputenc`. That should not pose any problem except for Greek, for which there are specific solutions.

Now that you will use `utf8`, your files' proper encoding should also be encoded with an encoding which will not be more restricted than `utf8`, so that you do not loose information. As `utf8` is one of the most complete encodings, the best thing is to save your files using `utf8`. It can be achieved under Microsoft Windows ©, and this is the default locale for Linux. If you are under Microsoft Windows ©, and that you do not encode your files using `utf8`, you might have troubles, but as, generally, people type documents in either their native language, or English, this should not be that problematic.

Keep in mind that we only saw an overview of these problems, and that specific languages might be problematic, especially for minorities. We did not give a complete recipe to make everything work, but these guidelines should foster a better comprehension of this problem.

This page intentionally left blank.

## L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> and Plagiarism

*Plagiarism* is a common issue. It is defined as (1995 *Random House Compact Unabridged Dictionary*, 1995)

The use or close imitation of the language and thoughts of another author and the representation of them as one's own original work.

I will here take a very concrete case: my case. This year, I had a group project to do, and the two other students did not contribute at all to the work. As I had to share my work with them because there was an oral exam and that the professor wanted it to be shared, I accepted to share it, but with a big watermark.

I had not realized that this choice would have been critical. Some days after, I realized that one of the two other students wanted to appropriate the work, and thereby claim its honesty and investment in the work. He then tried to remove the watermark, and, despite of many research, he never found how to remove the watermark. However, he could have done it. I learnt many things thanks to this situation, and I will here explain what I learnt on a T<sub>E</sub>X point of view.

My first reaction to ensure security was to **secure the PDF by using free tools**. This was a good *dissuasion*, as the two other students were stopped by this measure. By ‘securization,’ I here mean that I theoretically prevented the user from printing, selecting, or extracting content from the PDF file. However, PDF securisation like this is not widely accepted by PDF readers. Here is what Jay Berkenbilt (`qpdf`’s author) once told me.

‘The PDF specification allows authors of PDF files to place various restrictions on what you can do with them. These include restricting printing, extracting text and images, reorganizing them, saving form data, or doing various other operations. These flags are nothing more than just a checklist of allowed operations. The PDF consuming application (`evince`, Adobe Reader, etc.) is supposed to honor those restrictions and prevent you from doing operations that the author didn’t want you to do.

The PDF specification also provides a mechanism for creating encrypted files. When a PDF file is encrypted, all the strings and stream data (such as page content) are encrypted with a specific encryption key. This makes it impossible to extract data from without understanding the PDF encryption methods. (You couldn't open it in a text editor and dig for recognizable strings, for example.) The whole encryption method is documented in the specifications and is basically just RC4 for PDF version 1.4 and earlier, which is not a particularly strong encryption method. PDF version 1.5 added 128-bit AESv2 with CBC, which is somewhat stronger. Those details aren't really important though. The point is that you must be able to recover the encryption key to decrypt the file.

Encrypted PDF files always have both a user password and an owner password, either or both of which may be the empty string. The key used to encrypt the data in the PDF is always based on the user password. The owner password is not used at all. In fact, the only thing the owner password can do is to recover the user password. In other words, the user password is stored in the file encrypted by the owner password, and the encryption key is stored in the file encrypted by the user password. That means that it is possible to entirely decrypt a PDF file, and therefore to bypass any restrictions placed on that file, by knowing only the user password. PDF readers are supposed to only allow you to bypass the restrictions if you can correctly supply the owner password, but there's nothing inherent about the way PDF files are structured that makes this necessary.

If the user password is set to a non-empty value, neither `qpdf` nor any other application can do anything with the PDF file unless that password is provided. This is because the data in the PDF file is simply not recoverable by any method short of using some kind of brute force attack to discover the encryption key.

The catch is that you can't set restrictions on a PDF file without also encrypting it. This is just because of how the restrictions are stored in the PDF file. (The restrictions are stored with the encryption parameters and are used in the computation of the key from the password.) So if an author wants to place restrictions on a file but still allows anyone to read the file, the author assigns an empty user password and a non-empty owner password. PDF applications are supposed to try the empty string to see if it works as a password, and if it does, not to prompt for a password. *In this case, however, it is up to the application to voluntarily honor any of the restrictions imposed on the file.* This is pretty much unavoidable: the application must be able to fully decrypt the file in order to display it.

None of this is a secret. It's all spelled out in the PDF specification. So encrypting a PDF file without a password is just like encrypting anything else without a password. It may prevent a casual user from doing something with the data, but there's no real security there. Encrypting a PDF file with a password provides pretty good security, but the best security would be

provided by just encrypting the file in some other way not related to PDF encryption.’ [3]

Thus, one might not want to rely only on this PDF feature, especially if the desire is to set attributes without a password (see the italic sentence in the cited text).

My second idea was to *put a watermark on every page of the document*. For this, I used the `xwatermark` package, because I had not time to look for another way to achieve it; I was near the work’s due date.

I then compiled the `tex` document, secured it, and sent it.

In this series of practices, I should have realized that these two protections could totally be circumvented in an easy way. I knew it, partially, but had not much time to think about it. One needs to realize that such practices are not infallible: they are only dissuasive ways to prevent your work from being plagiarized.

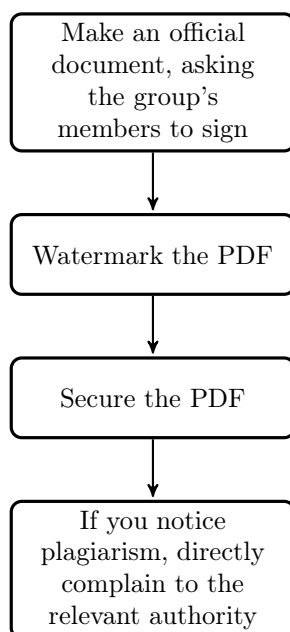
Let’s take, for example, the PDF security. One could simply run `pdf2ps`, and then `ps2pdf` on the resulting PS file, to recover exactly the same PDF without the security attributes (or hyperlinks). Thus, by using two commands, you can easily remove the PDF protection (assuming it involved only attributes, not passwords).

Next, there is the watermark that was L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>-inserted. There are different programs, especially for Windows users, that can remove watermarks. I tried them on my watermark, and none could remove it. Good point, but that does not mean that there is no program which is able to remove it. I might have forgotten one, or simply, a commercial program might be capable of this. (I never test commercial programs.) But I had committed an important mistake in my L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> watermark. The watermark is written on a different ‘PDF layer’ (one can conceive a PDF as being constructed from different layers which are superposed in some way) and is thereby not completely incorporated in the core document. Thus, if you use a L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> package to write a watermark in a document, do not forget to **merge layers**. This can be achieved easily. For example, using `lpr` under Linux, you can re-print the PDF document (with the different PDF layers) as a PDF document where the watermark and the text layers are merged together, thus making the differentiation between both really complicated for a ‘normal’ user. This can be achieved with a GUI too, evidently. For me, I can directly choose this virtual printer, and to print in a file, through my GNOME’s printing interface.

But one needs to keep in mind that all these measures are here as a source of dissuasion only. For example, whatever the protection, one can still take screenshots of the PDF’s viewer’s window, to make copies of the PDF content. This is a tedious way to do it, but if one wants it, he can do it. If even these screenshots were to be impossible to take, he could take a camera, and take pictures of his screen. All the measures you need to take against such behavior need to be adapted, and correlated in regards to other’s motivation to exploit your work. This is a very important point, as, once the work is sent, you cannot modify what you gave.

Another point which could be exploited is the use of a somewhat legal document, contraining the group's members to sign.

The best thing is evidently to avoid these problems, by talking together, as Humans are equipped with an extraordinary ability to share their feelings and to express themselves; however, communication problems sometimes arise, and, in this case, you might think about the aforementioned tricks. Here is thus what I suggest you to do, if such a problem arises (the first arrow being to follow if communication is somewhat broken):



We will now detail two PDF problems: search and cut-and-past, with their respective solutions. One might generate a PDF document using L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>, but if this document contains accented characters, searching for words containing accented characters could always fail, depending on the PDF reader.

The first problem is that the default font encoding is OT1. It should generally deal correctly with basic ASCII characters and the PDF operations on it. But once you will use accented characters, it might cause troubles with some PDF readers.

A solution is to use the T1 font encoding, conjointly with the `cmap` package. As a result, the

```

% ...
\usepackage[T1]{inputenc}
% ...
\usepackage{cmap}
% ...

```

directive should fix search and cut-and-paste in traditional PDF readers, even for Greek symbols in formulae. [\[87\]](#)

This page intentionally left blank.

## Turing Completeness

L<sup>A</sup>T<sub>E</sub>X is a Turing-complete programming language. Roughly, that means that the rules followed in sequence on arbitrary data can produce the result of any calculation. [187] That is, as one can do a lot of things with Turing-complete languages, you can do a lot of things with L<sup>A</sup>T<sub>E</sub>X, and, more generally, ‘compute everything that can be computed’ (but that does not imply that every implementation for every computation is simple!).

You might find an implementation of a Turing machine in L<sup>A</sup>T<sub>E</sub>X at [66]. This is rather theoretic, and I prefer showing the results of a Mandelbrot fractal generated by L<sup>A</sup>T<sub>E</sub>X. To avoid any ink wastes, consider having a look at [159] by your means. I here give the example of [173]. Consider the code

```
\newcount \Re \newcount \Im
\newcount \Zr \newcount \Zi
\newcount \Zrr \newcount \Zii \newcount \Ind

\newcommand{\MandIter}{%
  \divide \Zr by 16 \divide \Zi by 16
  \Zrr=\Zr \multiply \Zrr by \Zrr %
  \divide \Zrr by 256
  \Zii=\Zi \multiply \Zii by \Zi %
  \divide \Zii by 256
  \multiply \Zi by \Zr \divide \Zi by 256
  \multiply \Zi by 2 \advance \Zi by \Im
  \Zr=\Zrr \advance \Zr by -\Zii \advance %
  \Zr by \Re
  \let\next=\MandIter
  \count4=\Zrr \advance \count4 by \Zii
  \ifnum \count4>262144 \let\next=\relax \fi
  \ifnum \Ind=15 \let\next=\relax \else %
  \advance \Ind by 1 \fi
  \next
}
```

```

}

\newcommand{\MandLoop}{%
  \Re=\count0
  \multiply \Re by 196608 \divide \Re by \count2 %
  \advance \Re by -131072
  \Im=\count1
  \multiply \Im by 150000 \divide \Im by \count3 %
  \advance \Im by -75000
  \Zr=\Re \Zi=\Im \Ind=0
  \MandIter
  \ifcase \Ind
    .\or .\or :.\or -.\or +.\or =.\or *.\or i.\or I%
    \or H.\or O.\or X.\or M.\or \#\or @.\or .%
  \fi
  \let\next=\MandLoop
  \advance \count0 by 1
  \ifnum \count0>\count2
    \newline
    \count0=0
    \advance \count1 by 1
    \ifnum \count1>\count3
      \let\next=\relax
    \fi
  \fi
  \next
}

\newcommand{\Mandel}[2]{
  \count0=0 \count1=0 \count2=#1 \count3=#2
  \MandLoop
}

% Document
% -----
\documentclass[a4paper]{article}

\setlength{\parindent}{0pt}
\setlength{\oddsidemargin}{0pt}
\setlength{\topmargin}{0pt}

\begin{document}
\ttfamily
\setlength{\baselineskip}{0pt}
{\small

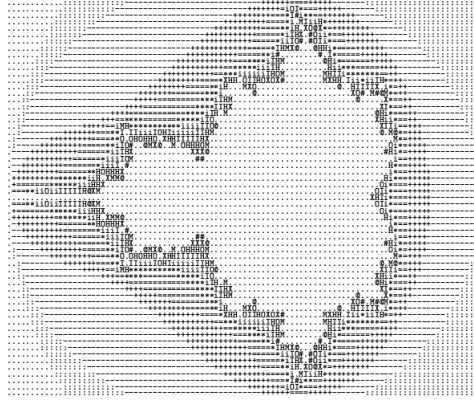
```

```

\Mandel{100}{60}
}
\end{document}

```

This code generates Figure 12.1.



**Fig. 12.1.** A Mandelbrot fractal generated using  $\text{\LaTeX}$ .

This page intentionally left blank.

## ConT<sub>E</sub>Xt, LuaT<sub>E</sub>X, teT<sub>E</sub>X, XeT<sub>E</sub>X

Beginners often hear these words. Now, what is the difference between these four concepts? We will here inspire from various sources such as Wikipedia [177, 178, 182, 186, 188] and [69, 122, 123] to give an extremely summarized description of these tools.

### 13.1 ConT<sub>E</sub>Xt

According to [177], ConT<sub>E</sub>Xt may be compared and contrasted with LaT<sub>E</sub>X, but the primary thrust of the two are rather distinct. ConT<sub>E</sub>Xt[177, 178]

- provides users easy and consistent access to advanced typographical control-important for general-purpose typesetting tasks,
- 's unified design avoids the package clashes that can happen with LaT<sub>E</sub>X,
- provides a multi-lingual user interface with support for markup in English, Dutch, German, French, and Italian and support for output in many languages including western European, eastern European, Arabic-script, Chinese, Japanese, and Korean.
- allows the user to use different T<sub>E</sub>X engines like pdfT<sub>E</sub>X, XeT<sub>E</sub>X, and LuaT<sub>E</sub>X without changing the user interface [122],
- is more monolithic than L<sup>A</sup>T<sub>E</sub>X,
- has a more homogeneous balise structure,
- has frequent upgrades,
- is more modular than L<sup>A</sup>T<sub>E</sub>X,
- retains L<sup>A</sup>T<sub>E</sub>X's structure-oriented approach [122].

ConT<sub>E</sub>Xt is not an engine, but more like a 'format' for T<sub>E</sub>X [122].

### 13.2 LuaT<sub>E</sub>X

As we saw in Section 13.1, LuaT<sub>E</sub>X is an engine. LuaT<sub>E</sub>X started as a version of pdfT<sub>E</sub>X with a Lua scripting engine embedded. After some experiments, it

was adopted by the pdfT<sub>E</sub>X team as a successor to pdfT<sub>E</sub>X [182]. The main objectives of the project are to [69, 182]

- provide a version of T<sub>E</sub>X where all internals are accessible from Lua,
- ensure downward compatibility,
- make a configurable variant of T<sub>E</sub>X,
- let the users write their own extensions instead of hard coding new features in T<sub>E</sub>X itself.

LuaT<sub>E</sub>X is an UTF-8 engine which is progressively becoming more and more interesting, but it is not yet finished [69].

### 13.3 teT<sub>E</sub>X

teT<sub>E</sub>X is a T<sub>E</sub>X distribution for Unix-like systems. teT<sub>E</sub>X is no longer actively maintained and its former maintainer Thomas Esser recommended T<sub>E</sub>X Live as the replacement. [186]

### 13.4 XeT<sub>E</sub>X

XeT<sub>E</sub>X is a T<sub>E</sub>X engine using Unicode and supporting modern font technologies such as OpenType or Apple Advanced Typography (AAT). Initially developed for Mac OS X only, it is now available for all major platforms. It [122, 188]

- natively supports Unicode and the input file is assumed to be in UTF-8 encoding by default,
- can use any fonts installed in the operating system without configuring T<sub>E</sub>X font metrics,
- can make direct use of advanced typographic features of OpenType, AAT and Graphite technologies such as alternative glyphs and swashes, optional or historic ligatures, and variable font weights,
- supports TrueType/OpenType fonts directly (where L<sup>A</sup>T<sub>E</sub>X's default fonts are Type 1). This can be harnessed by the powerful `fontspec` package which makes loading and using installed fonts really easy (sigh<sup>1</sup>),
- therefore aims at supporting languages.

A major difference between XeT<sub>E</sub>X and LuaT<sub>E</sub>X is that LuaT<sub>E</sub>X provides an extension mechanism, when XeT<sub>E</sub>X uses libraries. [69]

---

<sup>1</sup> Remember how it is generally difficult to do this in standard T<sub>E</sub>X, even with `fontinst`?

## Bib<sub>T</sub><sub>E</sub>X and Biber

An important part of this section comes from [132].

### 14.1 Bib<sub>T</sub><sub>E</sub>X Problems

The well-known Bib<sub>T</sub><sub>E</sub>X program is called to sort and format entries of your `.bib` files according to a specified style. This style is chosen according to the

```
\bibliographystyle{mystyle}
```

command, where `mystyle` is evidently the style you want to use (e.g. `IEEEtran.bst` to conform to the IEEE standards; [111]).

However, Bib<sub>T</sub><sub>E</sub>X suffers from many problems, which mainly come from the fact that this is an old program.

As an example, a recurrent problem is the fact that it chokes on non-ASCII database entries and forces you to write replacements like `{\ "a}` instead of `ä`. This is the major complaint about Bib<sub>T</sub><sub>E</sub>X, because apart from it being inconvenient, there are two major issues with this convention:

1. One subtle problem is that the extra set of braces suppresses the kerning on both sides of all non-ASCII letters,
2. Simply ignoring all accents may not be the correct way to handle them. For example, in Danish, the letter ‘`å`’ is the very last letter of the alphabet, so it should be alphabetized after ‘`z`’. Bib<sub>T</sub><sub>E</sub>X will sort it like an ‘`a`’. [63]

Bib<sub>T</sub><sub>E</sub>X sorting is also not case-insensitive (because its sorting algorithm uses ASCII codepage order). [63] This is generally of concern too.

### 14.2 Solutions

Many solutions were proposed to make a ‘better’ Bib<sub>T</sub><sub>E</sub>X. There is `bibtex8`, which includes support for 8-bit encodings (provided that you supply it with

a suitable `.csf` file and give the `--csfile` switch on the command line [63]), but still does not support Unicode [132]. It can sort case-sensitively. [63] There is also `bibtexu`, which should support UTF-8 bibliographies, but which is not well-known [132] and seems no longer maintained.

The state-of-the-art  $\text{\LaTeX}$  bibliography solution is said to be the `biblatex` package in conjunction with the `biber` program.

### 14.3 Differences Between Bib $\text{\TeX}$ and Bib $\text{\LaTeX}$

There are many advantages to use this solution. For example,

1. Encoding issues are gone thanks to Biber, so that you can output simple UTF-8 text without any escape sequences other than those mandated by the  $\text{\LaTeX}$  syntax (e.g. `\&`, `\#`, ...), [132]
2. Biber also lets you write more entries than with Bib $\text{\TeX}$  because it ‘has no resource limits at all,’ [174]
3. Bib $\text{\LaTeX}$  supports many fields that are not supported by many traditional styles (e.g., `doi`, `eprint` [132], `subtitle`, `titleaddon`, `maintitle` for multi-volume works, `editortype` [127], ...),
4. With Biber, you do not have to enclose title in another pair of braces too since it leaves the casing untouched (case insensitiveness), [132]
5. Bib $\text{\LaTeX}$  can make the process of writing ‘ibid,’ ‘op.cit.’ etc., when necessary, [26, 63]
6. Biber is in fact not limited to the `.bib` format, and it now has a modular driver architecture and will be extended to enable it to read other data sources. The latest Biber has beta support for RIS, for example. You can also mix and match as many data sources of any supported type too.

Bib $\text{\LaTeX}$  already supports the IEEE style. For example, you might choose this style using [16, 144]

```
\usepackage[style=ieee]{biblatex}
```

### 14.4 Migrating from Bib $\text{\TeX}$ to Bib $\text{\LaTeX}$

Fortunately, migrating from Bib $\text{\TeX}$  to Bib $\text{\LaTeX}$  is not a complex task. We shall consider the  $\text{\LaTeX}$  syntax point of view, and then the Bib syntax one.

#### 14.4.1 $\text{\LaTeX}$ syntax

Migrating from Bib $\text{\TeX}$  to Bib $\text{\LaTeX}$  should cause you no special troubles. Here are some transitional facts coming from [127]:

- With Bib $\text{\TeX}$ , the *bibliography style* was chosen using

```
\bibliographystyle{(somestyle)}
```

With BibL<sup>A</sup>T<sub>E</sub>X, as we said earlier, one can directly specify it at package's loading, together with other options:

```
\usepackage[style=(somebiblatexstyle),%
(other options)]{biblatex}
```

If you were using the `natbib` package, pass the `natbib=true` option to `biblatex`. The `natbib` option will automatically create the relevant aliases for the `\citep` and `\citet` commands, so you can use them as before. If your file has previously been compiled using `natbib`, you may need to delete some of the auxiliary files created by `latex` and `bibtex` for it to work properly,

- With BibT<sub>E</sub>X, *printing the bibliography* where you want is achieved thanks to

```
\bibliography{mybibfile}
```

where `mybibfile` is the name of your `.bib` file.

With BibL<sup>A</sup>T<sub>E</sub>X, use

```
\printbibliography
```

where you want your bibliography to be printed.

To specify your bib file, the command

```
\bibliography{mybibfile}
```

is generally used, so that both commands have separate roles.

#### 14.4.2 Bib Syntax

Changes to your BiBT<sub>E</sub>X `.bib` files are not mandatory, but you will miss some of the goodies offered by BibL<sup>A</sup>T<sub>E</sub>X if you decide not to modify them. It is advised at least to change [127]

- the `address` fields to `location` to be able to use the `maxitems` option,
- the `journal` fields to `journaltitle`,
- `year`, `month` and `day` to an ISO formatted date, e.g. `date={2010}` or `urldate={2010-08-11}`, to let BibL<sup>A</sup>T<sub>E</sub>X make use of some options like `date=short`, etc.

You might find the complete `biblatex` documentation at [63].

This page intentionally left blank.

## Using L<sup>A</sup>T<sub>E</sub>X Syntax as an Unambiguous Way to Communicate

A recurrent fact with everyday language is that it is subject to various interpretations, which sometimes lead to misconceptions and misunderstandings. This might not be a problem in special situations (some people use these facts in relationships, for example), but when one thinks about communicating on objective subjects, such as courses' material, it becomes detrimental.

When a student attends a course, he might understand, or not, what the lecturer says, for various reasons. He might also think that he has understood, when he has not really understood exposed concepts. Despite the fact that there might be many reasons to a (partial) misunderstanding of some given material, we here take an example: unprecisions due to (oral) language. More specifically, let us consider the case of a science lecturer whose course is based on mathematics.

Science courses require precision in their study, and thereby need to be well understood to be passed. Unfortunately, a lecturer has, as a matter of fact, no time to write all the mathematical expressions on the chart. A traditional way of explaining different things is the oral way: when something will not be written on the black board, it will be said. However, giving mathematical expressions orally can be difficult because of a lack of consistency in the oral expression of mathematical terms.

In some of these situations, it might thus be interesting to adopt a univocal language which can be fully understood provided people know its structure. This language can be L<sup>A</sup>T<sub>E</sub>X 'core' mathematical syntax.

For example, I would prefer hearing

*Let us consider  $\int_a^b f(x) dx$  [...]*

at the place of

*Let us consider the integral from a to b of  $f(x)$  [...]*

There are more obvious cases where the oral expressions are longer when used with an 'oral syntax,' than when explained using a L<sup>A</sup>T<sub>E</sub>X syntax. For me, using a L<sup>A</sup>T<sub>E</sub>X syntax allows your message to be univocal, and shorter. This

is pure benefit for lecturers and students (except that it might be considered as ‘too mechanical’ and ‘charmless’).

Another less disputable situation where it might be of interest, is when one needs to communicate by means of raw-text. Let us consider that you are a student and that you send a question (by e-mail) to a lecturer on some specific equations of some given course. You might send him to, say, ‘eqs. (3.109) and (3.115)’ but if the equations are not in his slides or book, or have changed since his (old) edition, you need to type them. You mainly have two choices:

1. If there are many equations you need to type, you generally make a `.tex` document, compile it, and send the resulting `.pdf` document,
2. If there are only some equations, one generally types them in the e-mail.

If you know the L<sup>A</sup>T<sub>E</sub>X syntax, why wouldn’t you use it? At the place of writing equations in some pseudo-language (which might be subject to various interpretations, and thus misunderstandings) resembling to L<sup>A</sup>T<sub>E</sub>X, you can directly type them using L<sup>A</sup>T<sub>E</sub>X syntax.

Even if the receiver is not L<sup>A</sup>T<sub>E</sub>X-ly aware, he might use syntax-to-graphical interpreters to understand your somewhat creepy equations.

My advice would be to sacrifice equations’ reading for equations’ graphical representation; that is, if your equations need `\left` and `\right` delimiters, and other more complex constructs, use them even if they decrease the human-readability of the equations’ code, because they will increase the human-readability of the equation’s graphical representation by a L<sup>A</sup>T<sub>E</sub>X engine.

There is evidently a compromise to do between unimportant (i.e. not useful in the context of the message) L<sup>A</sup>T<sub>E</sub>X syntax and core mathematical syntax, except if you are sure that your receiver interprets them using a compiler or a syntax-to-graphical interpreter.

The same method might be adapted with no difficulty when one needs to type some equations (s)he found after a long exercise. Once again, this helps having a shorter and univocal message. Moreover, you can directly embed your equations in forms, etc., after.

## Fonts' encodings

### 16.1 Do not Mix the Different Encodings

In my sixth article [84], I explained various aspects of the *input* encodings in  $\text{\LaTeX}$ , for ‘habitual’ dialects. Apart from the input encoding, one can also take care of using the ‘good’ *font* encoding. Some distinctions need to be done before going any further. Briefly,

1. A  $\text{\LaTeX}$  file is encoded in some encoding (for the file),
2. The file’s content is interpreted thanks to the `inputencoding`,
3. This content is translated into an output thanks to a font encoding.

### 16.2 Brief History

$\text{\TeX}$  (and  $\text{\LaTeX}$ ) traditionally used raster-graphic fonts produced by METAFONT for a specific device resolution. Until the arrival of Postscript, all applications used bitmapped fonts.

METAFONT was different because it used outlines to create the bitmaps and had parameters for optimising them. [23]

`dvips` originally produced PostScript files containing 300 or 600 dpi raster fonts, and so did the PDF files converted from that by e.g. `ps2pdf`. [57]

### 16.3 ‘Type’ Fonts

We first need to make a distinction between Type 3 fonts, that are bitmap (raster) fonts, and Type 1 ones, which are scaleable (vector). As a result, Type 3 fonts are considered resolution-dependent.

## 16.4 Bitmap vs Outline

In principle, given adequate resolution, the screen preview quality of documents set in bitmap fonts, and set in outline fonts, should be comparable, since the outline fonts have to be rasterized dynamically anyway for use on a printer or a display screen. [2] However, things are not that simple.

First, there are some problems because PDF viewers (especially Adobe Acrobat Reader, etc.) sometimes still do a rather bad job when displaying device-dependent Type 3 raster fonts: [57]

1. Texts in raster fonts can be displayed slow on the screen and with no or suboptimal anti-alias filtering. PDF viewers generally also do a poor job of downsampling high-resolution bitmap fonts to low-resolution screen fonts, [2, 57]
2. The 'Type3' raster fonts (previously) inserted by `dvips` lack information about which character each glyph represents, which interferes badly with full-text search and copy & paste. [57]

Second [110],

1. The Type 3 fonts are generated at a specified resolution, which is generally the printer's one. That is, typically 300 or 600 dpi are used,
2. Changing the resolution of bitmap fonts is no easy task and therefore the PDF readers produce terrible results when displaying these fonts on the screen,
3. Printing quality can be quite deceiving if the printer's resolution does not match the one of the bitmap font.

As a result, Type 3 fonts might appear as 'bad.' They are however sometimes unavoidable: there are still many font files of useful or essential specialist symbols that are only available in METAFONT format. [23]

If you've got a recent  $\text{\TeX}$  distribution (later than 2005), `dvips` should already use resolution-independent  $\text{\TeX}$  (Type 1) fonts, by default. [57]

## 16.5 The OT1 Encoding

Using OT1, Don. E. Knuth's original text encoding [17] as an input encoding makes inclusions of Type 1 fonts in the PDF files. It might appear nice at first sight, as we saw that Type 3 fonts were 'bad,' but OT1 has three main problems:

1. When accented characters are required,  $\text{\TeX}$  creates them by combining a normal character with an accent because it is 7-bit and uses fonts that have 128 glyphs (and so do not include the accented characters as individual glyphs). As a result, while the resulting output looks perfect, this approach stops the automatic hyphenation from working inside words containing accented characters, [116, 179]

2. You cannot properly copy-and-paste such words from the output (DVI/PS/PDF). That is, copying/pasting a non-ASCII glyph is impossible without having the glyph split up into its components, [116]
3. Characters like the pipe sign, less than and greater sign give unexpected results in text. [116]

Computer Modern (CM) OT1 fonts are hardly used in raster form, even if they are frequently available in this form. [18]

## 16.6 Hinting

Preliminary definition: *Hinting* is, briefly, ‘information which goes to the rasterizer along with the glyph outlines, and is used to make better decisions about which pixels to turn on.’ [64]

## 16.7 The Cork Encoding: T1

### 16.7.1 Description

As a result of OT1’s problems, T1 was defined as a new encoding: the Cork encoding. [17, 50] T1 is used with fonts that use 8-bit encoding. 256 glyphs are possible. [115, 116]

This encoding has been realized in a series of fonts designed with METAFONT, in at least one font series that is available both in Adobe Type 1 format and in OpenType format. [17]

The two encodings (OT1 and T1) are e.g. described at [45, 46].

### 16.7.2 Together With a Font Package

The T1 encoding does not necessarily uses Type 1 fonts, because the standard Computer Modern fonts have no *free* ‘European’ Type 1 equivalent for the T1 font encoding. As a result, some font packages to ‘use’ conjointly with T1 are generally proposed [60]:

- **lmodern**, Latin Modern fonts: Type 1 converted from METAFONT sources of Computer Modern fonts, [49]
- **cm-super**, Computer Modern Super fonts: Type 1 converted from METAFONT sources of various Computer Modern font families. [170]

The CM-Super family is very large and therefore difficult to maintain. [49] Both [114]

1. are derived from Computer Modern,
2. have problems with hinting.

There are some key differences between both [114]. For example, `lmodern`

- has a handmade vectorization,
- has revised metrics,
- provides more glyphs, especially diacritical characters,
- 's development goes on,
- is available in the same optical sizes as CM,

Also, `cm-super`

- is a vectorization of CM bitmap fonts but mainly automatically done,
- is available in more optical sizes (this 'feature' of `cm-super` switched is off by using the `fix-cm` package (highly recommended)),
- is recommended to be used conjointly with the `fix-cm` package to fix a lot of broken design decisions in `cm-super` (and in addition this makes the final PDF a bit smaller),
- contains Cyrillic fonts and `lmodern` doesn't.

Once installed, these packages do not need to be `\usepackaged`, and directly take part into the game.

Another package is sometimes proposed: `ae`. But despite giving a good-looking PDF,

1. using non-ASCII characters will break Copy & Paste in the PDF output, [114]
2. these fonts do not contain the guillemets, which are necessary for French. As a result, the `aeguill` package has been created to help you. [107]

There is a big difference between `ae` and the two aforementioned packages: `ae` is a *virtual* font package. That is, it allows us to map "bits of DVI file" to single characters in the virtual font; so we can create an "é" character by recreating the DVI commands that would result from the code "\'e." However, since this involves two characters being selected from a font, the arrangement is sufficient to fool Acrobat Reader, and that results in the copy & paste problem cited before with `ae`. If you can live with this difficulty, virtual fonts are a useful and straightforward solution to the problem. [104, 167]

## 16.8 Type 1, Type 3, TrueType and OpenType

Both Type 1 and Type 3 fonts are becoming obsolete if you do not use pure  $\text{\LaTeX}$ : OpenType is more or less the only modern font format that is widely used (there are also AAT and Graphite and possibly others). All input encoding, font encoding and rendering issues vanish once you use a modern engine (Lua $\text{\TeX}$ , Xe $\text{\TeX}$ ) and OpenType fonts. [17] Such engines effectively let you choose an OpenType font. [14, 17, 42]

Type 1 is the default for regular  $\text{\LaTeX}$ . [17] That does not mean that it is impossible to use OpenType fonts in  $\text{\LaTeX}$ : take a look at [48, 100, 101] for example.

OpenType supersedes TrueType, but installing TrueType fonts under L<sup>A</sup>T<sub>E</sub>X is possible too; see for example [32, 44, 68, 105, 160]. Using TrueType fonts is also possible under e.g. XeT<sub>E</sub>X. [151]

## 16.9 PDFT<sub>E</sub>X

PDFT<sub>E</sub>X can use OpenType or TrueType ([5, 6, 105]) fonts, but that requires TFM files (only XeT<sub>E</sub>X and LuaT<sub>E</sub>X can load OpenType fonts directly without the help of TFM files).

XeT<sub>E</sub>X and LuaT<sub>E</sub>X can all read Type 1, TrueType and OpenType fonts. PDFT<sub>E</sub>X built-in deals with Type 1 fonts too. [18]

More generally, pdfT<sub>E</sub>X is built so that a source must be available for all fonts used in the document, except for the 14 base fonts supplied by Acrobat Reader (Times, Helvetica, Courier, Symbol and Dingbats).

It is possible to use METAFONT-generated fonts in pdfT<sub>E</sub>X, but it is strongly recommended not to use METAFONT fonts if an equivalent is available in Type 1 or TrueType format, if only because bitmap Type 3 fonts render very poorly in Acrobat Reader.

Given the free availability of Type 1 versions of all the Computer Modern fonts, and the ability to use standard PostScript fonts, most T<sub>E</sub>X users should be able to experiment with pdfT<sub>E</sub>X. [158]

## 16.10 Practical Considerations

Nowadays, dvips grabs outline fonts automatically, and the PDF output at the end of the routine thus suffers from no ‘zooming problem’ as you would expect in significantly old distributions relying on -P command to dvips to grab outline fonts. [17]

With ‘standard’ languages and recent distributions, there should be no troubles when specifying no font encoding, and Type 1 fonts should be used directly and naturally at compilation. As a result, your documents would not suffer ‘zooming problems,’ but, as the default encoding is OT1 if you do not specify one, the three main problems given at Subsection 16.5 will arise. To solve this problem, it might be of interest to use a T1 font encoding:

```
\usepackage[T1]{fontenc}
```

We know that the problem of this approach is that it will make extensive use of Type 3 fonts, and thus create zooming problem, because fonts are replaced by ‘European Computer Modern’ Type 3 fonts.

But if either **lmodern** or **cm-super** is installed on your L<sup>A</sup>T<sub>E</sub>X distribution, Type 1 fonts will take on. If none of these packages has been installed, you will see zooming problems, and that will be an indicator: you then need to install one, preferably **lmodern** for dealing with Computer Modern-like fonts.

Evidently, you might try another font package, such as `palatino`, which will result in the inclusion of Type 1 fonts, that is, what you were expecting. [50]

The important thing is thus to use a T1 encoding together with Type 1 fonts. Now, you can choose a font of your taste. Have a look at e.g. [162].

Same inclusion remarks generally apply for PDF<sub>T</sub>E<sub>X</sub>. (For an example, take [25].)

### 16.11 Converting .ps Files That Were Produced With CM Bitmap Fonts

If you want to convert to PDF historic PostScript files that were produced with Computer Modern bitmap fonts, try the `pkfix` tool to replace these fonts in the PostScript with their Type 1 equivalents. [57]

### 16.12 Accents Encoding

Many people stick with writing accented characters such as `é` using `\'e`. It is important to know when this needs to be done, and when it is useless. Consider that your file is encoded using a proper encoding, i.e. 'an encoding that covers the set of characters you wrote in it.'

Now consider the following examples, assuming `lmodern` is installed on your computer:

1. `\documentclass{minimal}`  
`\usepackage[T1]{fontenc}`  
`\begin{document}`  
`é`  
`\end{document}`
2. `\documentclass{minimal}`  
`\usepackage[utf8]{inputenc}`  
`\usepackage[T1]{fontenc}`  
`\begin{document}`  
`é`  
`\end{document}`
3. `\documentclass{minimal}`  
`\usepackage[utf8]{inputenc}`  
`\begin{document}`  
`é`  
`\end{document}`

Only item 1 will result in a problematic output for evident reasons. There will not be any problem in item 2, and item 3 will make the document use OT1

as a font encoding with Type 1 fonts (which is bad because of the OT1 font encoding, but placed here for demonstration).

However, please note that the `\'` and other schemes are also used to overcome files' encoding clashes through OSes.

### 16.13 Summary

I personally learnt a lot writing this. As a result, here is a quick summary of what you need to keep in mind after having read this section. These remarks apply for `latex`→`dvips`→`ps2pdf`, and for `pdflatex`.

1. Type 1 fonts are nice, and so are OpenType fonts, because they are 'vector fonts,'
2. OT1 encoding always use Type 1 fonts, but OT1 suffers from three main drawbacks (see [16.5](#)),
3. T1 encoding should be preferred, but T1 automatically loads Type 3 fonts. As a result, you need to install font packages.

### 16.14 Internationalization

Note that the above text is only valid for L<sup>A</sup>T<sub>E</sub>X-'standard' languages: refer to specialized literature for arabic, cyrillic, slovak, etc., scripts.

### 16.15 Documents as a Source of Information

When one publishes a document on the Internet, e.g. using the PDF specification, it is often desirable for this file to be publicly searchable. That is, search engines should be able to parse your document content, whatever its language. They actually do pretty much the same job as what your PDF viewer does when you use the 'search' option, for e.g. some word.

It is known that for your document to be correctly interpretable by a PDF viewer, or, more generally, by a search engine, it needs to be correctly decoded. For languages without accents or special characters (which is a vague expression), you might thus refer to the preceding discussion.

This page intentionally left blank.

## A Student Point of View on the Aesthetics of Publications, from the Perspective of Effectiveness

Writing a document is a difficult task because the writer needs to think about two different concepts: the appearance of the document, and its content. Both constitute the message the writer wants to transmit.

We will here treat about the aesthetics of publications, from the perspective of effectiveness (which implies clarity). We begin by discussing some (but not all!) basic principles on how to write a good article, then treat about what L<sup>A</sup>T<sub>E</sub>X can help you to do to improve your document's content. We end with what you might do and do not about graphical æsthetism, and then end with some tips on how to ensure æsthetism, together with effectiveness, when writing an article.

### 17.1 Introduction

*“The only mistakes in typography are things done in ignorance.”* [153]

Writing a document is a difficult task because the writer needs to think about two different concepts: the appearance of the document, and its content. Both constitute the message the writer wants to transmit.

We will here treat about the aesthetics of publications, from the perspective of effectiveness (which implies clarity). We begin by discussing some (but not all!) basic principles on how to write a good article, then treat about what L<sup>A</sup>T<sub>E</sub>X can help you to do to improve your document's content. We end with what you might do and do not about graphical æsthetism, and then end with some tips on how to ensure æsthetism, together with effectiveness, when writing an article.

## 17.2 What You Need to do to Write a Good Article

We here consider that the reader is familiar with L<sup>A</sup>T<sub>E</sub>X. Evidently, using L<sup>A</sup>T<sub>E</sub>X is not a necessary condition for a good work, but it might simplify your task to some extent.

### 17.2.1 Use L<sup>A</sup>T<sub>E</sub>X

L<sup>A</sup>T<sub>E</sub>X's first role is to help the writer to focus more on the *content* of the document than on its appearance. 'Other writing programs' can sometimes lead to inconsistencies in the general presentation and layout of documents (if bad choices are done, for example), and the number of such inconsistencies generally rises with the document's length: it is more difficult to have a global view of a document (and thus on its presentation) when the document is quite big.

The appearance should not go against the message. The choice of the appearance of a document is conditioned by the message it tells you: you will not use the same styles for an advertisement or for a scientific paper.

As a result, if L<sup>A</sup>T<sub>E</sub>X is correctly used, the 'default' layout of a L<sup>A</sup>T<sub>E</sub>X document should inspire a clear presentation of concepts. If each element of the document is presented clearly, the resulting document should appear clear too. We will now discuss how you can make the content of your document clear. This is an important concept because every scientific document should be as clear as possible. This is not the case in some advertisements, for example, where mysteries might be artificially stirred up to create some special effects on the reader.

Using L<sup>A</sup>T<sub>E</sub>X correctly needs you to structure your work. That is, you need to use `\part`, `\section`, and other hierarchic elements. This is, for me, the most difficult part of any work: the structuration.

It is generally said that 'rules' are better understood when counterexamples are given. Consider thus texts that you found difficult to understand because they were lacking clarity. There are no general 'guidelines' that one can follow to write a good document, but one can learn a lot from reading poorly-presented or poorly-written documents.

### 17.2.2 Structure

Why was the message unclear? Because of a *lack of structuration*? Were such documents missing sections, subsections, ..., paragraphs, enumerate lists? Don't repeat these mistakes if you hated reading these documents. A reader should understand what your book speaks about when reading the table of contents (ToC).

Remember that the structuration step is a difficult one, mostly because it asks you to categorize concepts, and thus to compare their similarities to find the relations between every concept.

We will here detail some ways to write a clear document. A clear document is a document which is concise, precise, adapted, and univocal. There might be some other aspects to take into account, but these are the most important ones. We will develop them one by one.

### 17.2.3 Write Concisely

Let us now consider that you structure your work correctly using L<sup>A</sup>T<sub>E</sub>X. Now, one might have a tendency for writing much for saying ‘nothing interesting.’ This is generally not a good option for a scientific paper, because it needs to be *concise*. As a result, ensuring that the document conveys the information you wanted it to convey, in the *most direct way*, is an important, but a tedious task: one generally needs many reviewings to get a correct idea of his text and to be sure that some elements are not necessary.

Note that you can evidently reformulate or paraphrase, from time to time, some preceding concepts, but do it if that worths it: if your structure is already clear, and that your reformulation will contain exactly the same elements as elements which are already appearing clearly, there is no need to repeat them. If, on the opposite, there are some concepts whose roles are not highlighted from the text’s structure, first ask you if there is no structure problem in your text. Then, if necessary, repeat them, but knowingly.

### 17.2.4 Write Precisely

Now, *all the words* that are used look necessary. But are they *adapted*? That is, are they *sufficiently precise*? Bad choices of words is sad, because people only read what you wrote, not what you thought. As a result, if you used a more general word for a specific concept, it is possible that they never make the connection with the precise concept you were thinking about when using the more general word.

### 17.2.5 Write Adapted

Now that you are sure that they do not look ambiguous, confusing, or simply unprecise, *are they adapted to the public*? Is the article targetted for the audience, or are you the only one who can actually understand it fully? It is a shame to realize that some authors are more interested in demonstrating their erudition than in presenting information clearly to his reader.

Writing an article for a cutting-edge journal in some specific research field is not the same as writing an article for a less ‘aware’ public. This verification is often omitted, and, to be sure that your text is adapted, try to find readers of the same level of knowledge as your article’s potential readers, and ask them if the article sounds too difficult, or not.

Moreover, do not forget that being able to vulgarize some text if necessary without implying (too much) errors, but only deficiencies, is also an important

scientific skill. Less knowledgeable people in your field might be interesting for your research too.

### 17.2.6 Write Univocal

Another important aspect to consider is the univocity of interpretations. Keep in mind that if one interprets a word in one sense, another might interpret it differently. Such interpretations generally depend on one's past and experiences.

As a result, global (general) words might have different meanings for different persons. If you really need to use global words, refer to the dictionary to be sure that you use the *right word for what you mean*. You will then *use univocal sentences*.

### 17.2.7 Correct, but Later

After having worked a long time on the article, a common mistake is to re-read it *directly* to find what needs to be changed. If that works for obvious errors, and is morally comforting, it is generally better to *let an article 'sleep' and then re-read it with 'different eyes.'* If you cannot do this objectively, it is generally possible to ask other persons to do it. Be sure to re-read it with a totally different mind, just as if you were a potential reader or reviewer.

## 17.3 What L<sup>A</sup>T<sub>E</sub>X Brings You

### 17.3.1 Structure

We saw in the previous section that L<sup>A</sup>T<sub>E</sub>X gives you the opportunity of *structuring your work*. This is an essential element, and a necessary condition for a good work. However, we saw that this condition is far from being sufficient, and that, even if L<sup>A</sup>T<sub>E</sub>X helps you to keep the same layout all over the document, it does not make the work at your place!

### 17.3.2 Automatization

Another key feature of L<sup>A</sup>T<sub>E</sub>X, which misses from any other 'document editor,' is that it is a *programming language*. On the one hand, that means that it might appear as unintuitive when you learn it at the beginning, but, on the other hand, that means that this is a *good automatization tool*.

Consider that you have tasks that need to be repeated, be it at every section, or every (odd, even, both) page, etc. Then, with little knowledge, you can *automatize such tasks*. There are many concrete examples:

- Asking the reader to 'turn the page' on every odd page (except the last; see [85] for this),

- Automatically writing a specified text when you use a given command (such as using `\R` for  $\mathbb{R}$ , thus preventing you from sometimes using  $\mathcal{R}$  or  $\mathbf{R}$  for the Reals),
- Automatically executing a special command when specific criteria are fulfilled, etc.

More globally, a  $\text{\LaTeX}$  document can entirely be personalized, at least to some extent, but this personalization can be conserved through all the document, which means that it adds consistency to your document.

### 17.3.3 Dynamic Referencing

$\text{\LaTeX}$  also offers you dynamic references: you can send the reader to a given page according to the text present on the page, and not on its number. That means that you do not need to think about page numbers. Introduce references:

`\label{sec:myfirstsection}` where you want, and then cite them:  
`\ref{sec:myfirstsection}` where you want them to be cited.

## 17.4 Graphical Aesthetism

### 17.4.1 Why Not

Clear articles can rhyme with graphical aesthetism too. Some well-written documents sometimes *look* boring, because they are not sufficiently ‘pretty.’ Having a clear, concise, precise, and interesting document is nice, but if it is *readable*, that is even *better*.

### 17.4.2 But Not Too Much

However, there are *some traps* one needs *not to fall into when prettifying documents and using graphics*. We here give some tips on how to use some kinds of drawings according to the data they represent, or the information they should convey. Most of this comes from [153, 161], sometimes literally. If you are interested in reading more, consider looking at [161].

#### Example: 2D pie Charts are Okay, But 3D’s are Evil

Consider the case of pie charts, for example. Many authors make an extensive use of ‘3D’ pie charts, i.e. pie charts with an impression of relief. Adding a relief dimension is interesting if there is a complement of information in it, or, generally, their pie charts could as well be drawn as simple circles.

Using 3D pie charts introduce a way to distract the reader, and also introduces ambiguity because the relatives sizes in a pie chart are very strongly distorted. Till Tantau even recommends to never use 3D pie charts, and qualifies them as ‘evil.’

### Some Tips on Drawings

More generally, my philosophy when prettifying documents is: prettify as much as possible to make the article readable, but do not overload, and, above all, do not introduce ambiguities (as in the pie chart example). *Prettifying* should always *serve your document*, and never go against it. Keep this in mind!

After having placed a drawing in a document, ask you if it

- a. does not introduce ambiguities (e.g. univocal),
- b. is really useful (i.e. nonredundant),
- c. conveys the information directly, or if it can be drawn differently to be understood in an easier fashion. Consider some data. There are some cases where bar charts are more interesting than pie charts, but there are also some cases where rough data is more interesting than bar charts,
- d. is minimal, i.e. do not draw complex graphics containing or conveying more data than what is needed for the explanation and for their present role,
- e. helps the viewer think about the information rather than the design,
- f. encourages the reader's eye to compare the data.

In a scientific document, drawings are generally not there for embellishment: they should serve the document's purpose. However, having beautiful graphics is better, and important because scientific documents' embellishment contributes to their readability.

### Some Tips on Graphics

Also think about these tips for graphics:

- a. Use as possible the same program and style for creating the graphics of an article (for consistentness' sake).
- b. Treat graphics as first-class citizens of your papers. They deserve as much time and energy as the text does. Indeed, the creation of graphics might deserve even more time than the writing of the main text since more attention will be paid to the graphics and they will be looked at first. Take a look at [157]: you will realize that graphics can be extremely informative, still being pretty.
- c. Do not scale graphics: when generating graphics using an external program, create them "at the right size."
- d. Use the same font(s) both in graphics and the body text.
- e. Almost all graphics will contain labels, that is, pieces of text that explain parts of the graphics. Do not forget to write axis' names, even if the graphic appears trivial. What might be trivial to you might be totally hard to understand for somebody else. When placing labels, stick to the following guidelines:
  - a) Follow the rule of consistency when placing labels. You should do so in two ways: be consistent

- i. with the main text, that is, use the same font as the main text also for labels;
  - ii. between labels, that is, if you format some labels in some particular way, format all labels in this way. In addition to using the same fonts in text and graphics, you should also use the same notation. For example, if you write  $1/2$  in your main text, also use  $1/2$  as labels in graphics, not 0.5. A  $\pi$  is a “ $\pi$ ” and not 3.141.
- b) Labels should be legible. They should not only have a reasonably large size, they also should not be obscured by lines or other text. This also applies to of lines and text behind the labels.
- c) Labels should be “in place.” Whenever there is enough space, labels should be placed next to the thing they label. Only if necessary, add a (subdued) line from the label to the labeled object. Try to avoid labels that only reference explanations in external legends. Readers have to jump back and forth between the explanation and the object that is described.
- d) Consider subduing “unimportant” labels using, for example, a gray color. This will keep the focus on the actual graphic.
- f. Use the same line width in text and graphics.
- g. The “line width” for normal text is the width of the stem of letters like T. For  $\text{\TeX}$ , this is usually 0,4 pt. However, some journals will not accept graphics with a normal line width below 0,5 pt. When using colors, use a consistent color coding in the text and in graphics. For example, if red is supposed to alert the reader to something in the main text, use red also in graphics for important parts of the graphic. If blue is used for structural elements like headlines and section titles, use blue also for structural elements of your graphic.  
However, graphics may also use a logical intrinsic color coding. For example, no matter what colors you normally use, readers will generally assume, say, that the color green as “positive, go, ok” and red as “alert, warning, action.” Consider this too.
- h. When you design a graphic, you should also eliminate everything that will “distract the eye.”
- i. At the same time, you should try to actively help the reader “through the graphic” by using fonts, colors, line widths to highlight different parts.

### Some Bulk Tips

- a. In pie charts, and in other drawings/schemes, try to associate ideas belonging to the same concept by using relative colors, and do not apply colors randomly. [153]
- b. A loosely-spaced grid is less distracting than a very closely-spaced grid.
- c. Dashed lines create many points at which there is black-to-white contrast. Dashed or dotted lines can be very distracting and, hence, should be avoided in general.

Do not use different dashing patterns to differentiate curves in plots. You lose data points this way and the eye is not particularly good at “grouping things according to a dashing pattern.” The eye is much better at grouping things according to colors.

- d. Background patterns filling an area using diagonal lines or horizontal and vertical lines or just dots are almost always distracting and, usually, serve no real purpose.
- e. Background images and shadings distract and only seldom add anything of importance to a graphic.
- f. Cute little cliparts can easily draw attention away from the data.

However, my point of view about little cliparts is that this is generally a good idea to use some specific pictograms such as ‘TIP’ or ‘PITFALL–DANGER’ like those found in some learning books. However, a different pictogram needs to be used for each kind of note, and the same pictogram needs to be used for the same notes.

### Some Figure Tips

For figures:

- a. Use interesting captions. A good caption adds as much context information as possible. While this information can also be given in the main text, putting it in the caption will ensure that the context is kept.
- b. Reference the graphic in your main text.

## 17.5 Writing Æsthetism

In the preceding section, we gave, among other things, some ways to use graphics correctly. However, pictures and drawings are not the only elements which contribute to a document’s æsthetism.

We know that L<sup>A</sup>T<sub>E</sub>X respects as much as possible standard typographic rules, and it thus generally gives a good-looking document. But once your document is structured, and that your text has been categorized in every hierarchic place of the document, there are still some textual elements you might want to work on.

### 17.5.1 Emphazing Something

For example, how do you *emphasize on something*? There are many different *wrong* ways to do it which are generally used by beginners.

In traditional text editors, emphasizing some portion of text is generally done by applying a ‘bold’ command so that what needs to be emphasized actually appears as bold text. This is totally disadvised for typographic

reasons. Some other users also underline what they need to emphasize on. This is also a bad idea, for typographic reasons too.

A better way to do it is to use slanted text, or italic shape. Do not forget to emphasize key parts of sections, subsections, ... It always looks clearer to me, and it should be the same for you. Take a look at what I emphasized in Section [17.2](#).

### 17.5.2 Choosing a Font

Nowadays, many fonts are available for L<sup>A</sup>T<sub>E</sub>X. When publishing in a journal, you have no choice, because the font is imposed by the editor. However, if you write an article for some friends, or anything where you are not obliged to stick with a predefined font, you might either stick up with the good old Computer Modern typeface (the default L<sup>A</sup>T<sub>E</sub>X (and T<sub>E</sub>X) one), but you could also try new fonts. For this, consider choosing a font that matches your context, and that looks ‘pretty’ to you. For example, you might decide not to use a ‘Comic Sans MS’ font when writing some article on a serious topic.

### 17.5.3 Using fancyhdr

Readers generally appreciate to know which part of your document they are currently reading. As hierarchic elements (such as what is after `\subsubsection{...}`) might contain a lot of text, it is thus a good idea to use a package such as `fancyhdr` so that (at least) current section’s title is displayed at some place in the top or bottom of the page.

This page intentionally left blank.

## eBooks

eBooks are progressively more and more used by people. There are many advantages in reading using eBooks. This does not put aside traditional books, evidently, but provides interesting opportunities depending on your lifestyle or simply your tastes.

We here describe some reasons to use eBooks, together with L<sup>A</sup>T<sub>E</sub>X-composed documents. We continue by explaining technical aspects that might improve and help you with your L<sup>A</sup>T<sub>E</sub>X eBook. We end with a short discussion about the PDF file format and its interest for the eBook type of documents. Writing a good content for the eBook does not enter in the scope of this article.

### 18.1 eBooks: Why?

We are here concerned with L<sup>A</sup>T<sub>E</sub>X eBooks. Some (traditional) books are written using L<sup>A</sup>T<sub>E</sub>X, and then printed. In some situations, it is of great interest to read an eBook at the place of reading a hard copy.

Most of L<sup>A</sup>T<sub>E</sub>X books are scientific or technical texts. As a result, many L<sup>A</sup>T<sub>E</sub>X documents are long, and thus

- heavy (to carry with you),
- bad for the environment if you print them,
- difficult to handle.

Moreover, you do not necessarily need to read an entire book in some cases.

Using eBooks gives you numerous advantages. For example, with an eBook, you can

1. quickly search and find some keywords in a text;
2. close and open it again easily;
3. do many other well-known things.

Our aim is here not to discuss the interests of using eBooks, which are already well-known, but to explain why eBook versions could be interesting for  $\text{\LaTeX}$  documents, and how they could be created.

## 18.2 $\text{\LaTeX}$ and eBooks

$\text{\LaTeX}$  offers many advantages over traditional typesetting, and its merits do not need to be explained again.

### 18.2.1 Why?

Many scientific books, proceedings, papers, and some romans are already printed ‘hard copy’ but composed using  $\text{\LaTeX}$ . As a result, electronic versions of the documents are generally available.

They are however rarely adapted to eBooks readers. Why wouldn’t we try to write  $\text{\LaTeX}$  eBooks, or at least eBook versions of our  $\text{\LaTeX}$  documents?

1.  $\text{\LaTeX}$  provides many interesting features to the reader: hyperlinks, glossaries, indexes, etc., that are easier to use when one deals with an electronic version of a document. Consider for example going to page  $x$ . Is not that simpler to click on the page number than to move a pack of sheets until you find the correct number?
2. You will see in this article that writing an eBook using  $\text{\LaTeX}$  is an easy task for a person who is familiar with  $\text{\LaTeX}$ ;
3. There is no reason to make  $\text{\LaTeX}$ -composed documents unavailable to eBook readers, especially if you know that they are progressively becoming ubiquitous.

It is of great interest to write  $\text{\LaTeX}$  documents that can be read on eBooks.

### 18.2.2 How?

However, it is not that technically easy, especially if you have never tried it. We all know that, at the beginning,  $\text{\LaTeX}$  was not created with eBooks in DK’s mind! That does not mean that making  $\text{\LaTeX}$  documents available to eBook readers is an impossible task, but that it will not be straightforward if you are not accustomed to it. (Just as many things under  $\text{\LaTeX}$ , but we all appreciate when the so-expected results appear in our documents.)

To give me an idea, I used a web engine and typed related keywords. Needless to say that I did not encounter many related results. But on-topic results were interesting. Among them, I found three topics on a forum: [89, 90, 91].

What are the biggest problems one needs to solve when considering an eBook audience at the place of a hard-copy audience?

1. Dimensions of eBooks are smaller;
2. eBooks are electronic.

When comparing to a computer audience, the biggest difference is that dimensions are smaller. An eBook reader does not need to scale your document down to some proportionality factor. *You* need to scale your document down to this factor. The following dimensions: 90 mm × 120 mm are generally considered, because majority of eBooks readers are offered a 6" diagonal, with 3 : 4 aspect ratio.

Now, what might improve the document's readability and effectivity? You want

1. to put 'as much text as possible' on each page of the document. However, 'as much text as possible' needs to be understood clearly: that does not mean you can forget every typographic rule! Use different margins if you feel the need to do it;
2. letters to be easy to read. That means that you might choose a new font;
3. text to be easy to read. Do not use long sentences: break them up into bulleted lists or divide separate thoughts into different paragraphs. Large blocks of text are difficult to read and do not have enough white space to give the reader's eyes a break [73];
4. things to be clickable: glossaries' entries, indexes, or any other `hyperref`-linked element.

A first minimal example was submitted at [91] by 'frabjous':

```
\documentclass[12pt,oneside]{book}

\usepackage[papersize={90mm,120mm},%
margin=2mm]{geometry}

\usepackage[T1]{fontenc}
\usepackage[charter]{mathdesign}

\usepackage[normalmargins]{savetrees}
\sloppy
\pagestyle{empty}

\usepackage{titling}
\title{The title}
\author{The author}
\date{\today}

\usepackage{hyperref}
\hypersetup{pdftitle={\thetitle},%
pdfauthor={\theauthor}}
```

```
% ...

\begin{document}
\maketitle
\tableofcontents

% ...

\end{document}
```

The most important thing one can notice is that this minimal example is extremely simple, yet it produces a simple but functional output.

It exploits margins to a maximal extent, and displays no page numbering. It thus allows each page to contain as much as possible material.

Another contributor (namely ‘ahi’) posted at [91] a translation of ‘The Art of War,’ a book originally written by Sun Tzū. The same person posted at [90] ‘The Complete Memoirs of Casanova,’ which I personally find L<sup>A</sup>T<sub>E</sub>X-ly readable.

Let us consider the code of ‘The Art of War,’ that was provided with the output. The whole code is not complex, especially what follows the preamble (the core of the document is L<sup>A</sup>T<sub>E</sub>X-ly simple). As habitually, the preamble is somewhat tricky, and some of its aspects need to be discussed:

- The class of the document is `memoir`, with `10pt` and `openany` options. The `openany` option makes chapters begin on the next page available. This helps having a short document containing as much as possible material on as little as possible pages. The advantage of the `memoir` class is that it automatically writes the name of the current chapter on each page, separated from the text by a rule. This helps the reader remembering the subject of what (s)he is currently reading (in case (s)he would have forgotten it!). As we will see on next page, the `memoir` class also lets you use different styles for titles.
- Fonts are also different:

```
\usepackage{kerkis}
\usepackage{yfonts}
```

- The `microtype` package is used.
- Margins are different too:

```
hmargin={0.17in, 0.17in}
```

and

```
vmargin={0.50in, 0.17in}
```

are used in the `geometry` package’s parameters, with the same `papersize` than in the minimal example provided here above.

- The following penalties are imposed:

```
\widowpenalty 3999
\clubpenalty 3999
```

This personally gives me a good impression. Note that `\clubpenalty` is the penalty for a broken page, with a single line of a paragraph remaining on the bottom of the preceding page.

- The title of the book is written on each page:

```
\makeevenhead{ruled}{\small%
\emph{\rightmark}}{\small%
\scshape The Art of War, Sun Tz\u{u}}
\makeatletter
\makeoddhead{ruled}{\small%
\emph{\rightmark}}{\small%
\scshape The Art of War, Sun Tz\u{u}}
```

- The following lengths are redefined:

```
\setlength\beforechapskip{0pt}
\setlength\midchapskip{5pt}
\setlength\afterchapskip{30pt}
```

- A chapter style is defined:

```
\makechapterstyle{plroman}{
\renewcommand\chaptername{}

\renewcommand\printchaptername{}

\renewcommand\printchapternum{%
\color{gray}\centering\MakeUppercase%
{\fontsize{1in}{2in}\selectfont%
\romannumeral\thechapter}}

\renewcommand\chapnumfont{\HUGE%
\rmfamily\centering\romannumeral}

\renewcommand\chaptitelfont{\Huge%
\centering\color{black}\vskip%
\midchapskip\vskip\midchapskip}

\renewcommand\afterchapternum{%
\par\nobreak\vskip\midchapskip%
\vskip%\midchapskip}

\renewcommand\printchapternonum{%
\vphantom{\chapnumfont \thechapter}
\par\nobreak\vskip\midchapskip%
```

```
\hrule\vskip\midchapskip}
}
```

This style is not necessary: the document stills looks great without it. This simply adds some personal touch to it, which does not harm. Another advantage of the `memoir` class is that you can use many different styles; see e.g. [70].

The document can be compiled using PDFLaTeX only.

Note that there are different ways to specify the global dimensions and the margins. The advantage of using the `geometry` package is that it automatically deals with your input in relation to the PDF output. That is, if you had specified

```
\textheight=120mm
\textwidth=90mm
```

at the place of using

```
papersize={90mm,120mm}
```

in `geometry`'s arguments, you would have ended with a PDF with a majority of white unused space, still with standard dimensions, but with a printed zone of 120 mm × 90 mm.

Some authors use `\sloppy` command, pretending this is the miracle solution to preventing overfull boxes (because of overfull lines). (You might have noticed that `\sloppy` was used in the minimal example given at the beginning of this article.)

Without it, L<sup>A</sup>T<sub>E</sub>X will truncate words that it cannot fit in a line due to its standards. [89] As said in [89], 'it is certainly less annoying to have extra whitespace in lines than having parts of words missing due to not fitting on the page.'

The problem with this command is that you get ugly justification and instead of using hyphenation, it will add white spaces most of the time. The best solution here is to change a bit those advanced settings for hyphenation. [89]

But how can you change these settings, and which settings would you change? We saw lately that penalties could be adapted to your needs. With greater mentioned penalties, L<sup>A</sup>T<sub>E</sub>X will more 'do its best' to avoid widows and orphans. However,

1. setting them to the maximum value of 10000 is evidently not the solution;
2. keep in mind that intermediary values do not have a high influence on the result.

The role of the `microtype` package is to deal with these problems too. (But either modify the penalties manually or let it try.)

### 18.2.3 Is PDF the Right Format?

Many eBooks readers are not capable of reflowing the text, hyphenate, and allow for font size changes. They can either unsupport this, or the PDF file they are dealing with might be ‘unreflowable.’ Reflowable attribute is a recent PDF capability, and is not supported in some PDF creators.

For this reason, PDF is sometimes considered as a bad format for eBooks, because its interest is to give a static representation of a document on the dimensional point of view. That is, the PDF specification was built with the ‘reproducing image’ capability in mind. For some [92], it is thus inherently inadapted to eBooks, and thus not an eBook format.

One of the problems is that the screen dimensions of the readers are not standard. As a result, if you provide an eBook whose size does not match the reader’s size, the reader will need to adapt the PDF output to its screen. If the eBook reader and the PDF file are compatible, there should be no problem, but having a compatible (i.e. reflowable) PDF file is not habitual. By default, neither the `ps2pdf` nor the `pdfLaTeX` routes give a reflowable PDF.

Ideally, one would need a reader with a  $\text{\TeX}$  engine that re-typesets the document every time the user changes the font size or orientation, which has obviously not been implemented. [34]

Many people find that ‘EPubs’ are better than PDFs on eBook readers, both in appearance and also in efficiency. With EPubs, if you have a separate html file for each chapter, only the current chapter needs to be loaded rather than the entire book. This thus provides many advantages. The EPub format is actually XHTML and the `tex4ht` package can generate XHTML. With a little post-processing, the output from `tex4ht` can be made into a `.epub` file. Since an EPub is XHTML, it is not bound to ‘pages’ at all and is implicitly reflowed, depending on the e-reader screen size. EPubs can be generated using `latex + tex4ht`. [34]

## 18.3 Conclusion

Despite critics on the PDF format, producing a PDF file for a known-by-advance screen dimension is sufficient to give a correct eBook composed using  $\text{\LaTeX}$ .

Several other solutions do exist, and can be studied if necessary. Meanwhile, it is possible to produce an eBook using  $\text{\LaTeX}$  by writing a simple  $\text{\LaTeX}$  document as it was shown before.  $\text{\LaTeX}$  can thus be used to produce eBooks, but under some restrictions. These restrictions might be overcome, if needed, by studying other solutions such as the `.epub` file.

This page intentionally left blank.

---

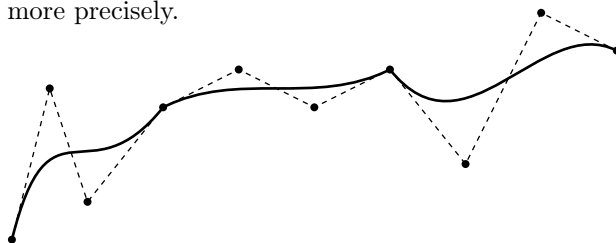
## L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> Learning Curve

L<sup>A</sup>T<sub>E</sub>X is clearly not the easiest language to learn, especially when one is used to WYSIWYG programs. Moreover, beginners sometimes do not see any interest in using it, and feel frustrated because they do not often get the expected results.

L<sup>A</sup>T<sub>E</sub>X learning curve is quite steep at some moments, and even steeper when you have never used it before.

My (little) experience tells me that the most difficult thing one encounters when explaining L<sup>A</sup>T<sub>E</sub>X to others is convincing them that ‘L<sup>A</sup>T<sub>E</sub>X is nice.’ By this, I mean that L<sup>A</sup>T<sub>E</sub>X has numerous advantages over traditional ‘typesetting’ programs that are generally not clear to the uninitiated.

Once you have convinced people that L<sup>A</sup>T<sub>E</sub>X is not only a good alternative, but simply *the* alternative for most uses (especially for mathematical typesetting), people do generally learn by themselves, because they know that it worths it. I thus describe L<sup>A</sup>T<sub>E</sub>X learning curve by the following Bézier curve (which shows the perceived difficulty versus time). Let us discuss its portions more precisely.



1. At the beginning, L<sup>A</sup>T<sub>E</sub>X seems like a strange thing, and is completely different from WYSIWYGs. Syntax looks complicated, and people do not really understand why they need to learn so much ‘commands’ for having a result *after* compilation. Memorization begins, but hardly. Compilation generally fails, but not necessarily! That is the moment where people wonder why they were obliged to learn this ‘awful’ language.

2. The new L<sup>A</sup>T<sub>E</sub>X user is progressively understanding the main L<sup>A</sup>T<sub>E</sub>X concepts, and is able to write simple documents, generally without any compilation error. Time spent at learning basic concepts now pays.
3. Things do not get quite more complicated as time goes by, because you generally try some things at your level of knowledge, to practice and give you self-esteem. Competence begins to install.
4. By practicing more and more, things end up by looking fairly easy, and you generally wonder why you did not begin learning it sooner.
5. Things sometimes get darker because you try to learn and understand more difficult things. Proportionally, you sometimes manage to learn things that look now as difficult as basic concepts were looking when you just began.

## 19.1 Most Common Beginners' Mistakes

What are beginners' most common mistakes? After having looked at different L<sup>A</sup>T<sub>E</sub>X codes, I established a list of the errors I encountered among sophomore friends' L<sup>A</sup>T<sub>E</sub>X documents. This list is constituted of 20 elements, and is not meant to be exhaustive:

- i. Writing mathematical function names without any `\`. That results in ugly outputs.
- ii. Using `\\` to separate paragraphs. That results in a typographically unacceptable output.
- iii. Using `\textit` or other commands to write some code. More generally, misunderstanding that L<sup>A</sup>T<sub>E</sub>X compiler will *interpret* the content of the `.tex` file if not enclosed properly. That results in weird compilation errors whose source is difficult to identify.
- iv. Misunderstanding L<sup>A</sup>T<sub>E</sub>X's interest: focusing on the document's content at the place of its appearance, thereby helping you to write faster better documents. I saw different persons taking hours to modify standard L<sup>A</sup>T<sub>E</sub>X typesetting parameters.
- v. Misunderstanding L<sup>A</sup>T<sub>E</sub>X's respect of typographic standards (or being unaware of any typographic standards): trying, again, to modify various parameters so that the document appears as 'better' than L<sup>A</sup>T<sub>E</sub>X's standard output.
- vi. Misunderstanding the global L<sup>A</sup>T<sub>E</sub>X extension system: things work by packages, and it is better to use e.g. the `align` environment than to `\hspace` some non-display math, for example!
- vii. Misunderstanding the difference between display and non-display maths.
- viii. Misunderstanding typography in general, e.g. using `"` for opening and closing guillemets, whatever the language (also mentioned at [8, 61, 99]).
- ix. Misunderstanding L<sup>A</sup>T<sub>E</sub>X math syntax: I saw different persons making their equations in WYSIWYG programs, then making a screenshot and embedding the screenshot as a Figure in the L<sup>A</sup>T<sub>E</sub>X document!

- x. Misunderstanding the automatization capability of L<sup>A</sup>T<sub>E</sub>X: at the place of using `amsthm` and

```
\begin{theorem}[My theorem.]
  This is true!
\end{theorem}
```

some users manually number their theorems, corollaries, remarks, ..., and manually format them. As a result, they lose any benefit of using L<sup>A</sup>T<sub>E</sub>X as an automatization tool, and tend to have a non-homogeneous output, e.g. sometimes writing

**Theorem 1 (My theorem).**

and sometimes using

**Theorem 1.** (My theorem)

or doing such things.

- xi. Misunderstanding image formats (also given at [61]).
- xii. Forgetting to use math mode: this is  $a+b=c$  not  $a+b=c$  (also mentioned at [8, 31]). Or using math mode when this is not appropriate.
- xiii. Forgetting to make spaces when needed (also mentioned at [8, 31]): this is (the d depends on your tastes)

$$\int_a^b f(x) \, dx, \quad \text{not} \quad \int_a^b f(x) dx \quad !$$

- xiv. Inverting `\caption` and `\label` in a `figure` or `table` environment.
- xv. Not caring about compilation errors, for editors that try to carry on when errors are encountered. That results in not profiting from different occasions to learn new L<sup>A</sup>T<sub>E</sub>X things.
- xvi. Mixing L<sup>A</sup>T<sub>E</sub>X warnings, errors, etc. Some users think that, as they received warnings during compilation, there is something fundamentally wrong in their code!
- xvii. Rejecting the fault over L<sup>A</sup>T<sub>E</sub>X compiler: it often happens to make typos (also given at [108]) such as

```
\tableofcotnetns
```

and, in such cases, you are so sure that you used the 'right' command (that you just read on some informational website) that L<sup>A</sup>T<sub>E</sub>X must be faulty!

- xviii. Using `<` and `>` at the place of `<` and `>` (this still happens with more trained writers).
- xix. Using `<<` and `>>` at the place of `<<` and `>>` (also given at [61]).
- xx. Inventing a totally new L<sup>A</sup>T<sub>E</sub>X syntax. If you decide to (re)invent some commands, let L<sup>A</sup>T<sub>E</sub>X know it, or do not do it (unless you want it to complain).

The `lacheck` utility has been created for the purpose of finding common mistakes in L<sup>A</sup>T<sub>E</sub>X documents. You might install it using your regular distro installer. Have a look at [\[74\]](#) for its manpage.

## Some Useful L<sup>A</sup>T<sub>E</sub>X-Related Webservices

It might be useful to use some specific L<sup>A</sup>T<sub>E</sub>X ‘webservices.’ Here is a small (and not exhaustive) list coming from [119] (except the Springer’s one which comes from my personal experience):

- *DeT<sub>E</sub>Xify* (<http://detexify.kirelabs.org/classify.html>) allows you to draw a symbol and it will then provide a list of likely L<sup>A</sup>T<sub>E</sub>X commands to create the symbol.
- *Uniquation* (<http://uniquation.com/>) is a math search engine that indexes and searches formulas represented in a T<sub>E</sub>X format.
- *Springer’s L<sup>A</sup>T<sub>E</sub>X Search* (<http://latexsearch.com/>) is in the same kind.
- *Google Docs* (<http://docs.google.com/>) supports L<sup>A</sup>T<sub>E</sub>X equations.
- *L<sup>A</sup>T<sub>E</sub>X Lab* (<http://docs.latexlab.org/>) supports full L<sup>A</sup>T<sub>E</sub>X inside Google Docs with realtime PDF preview.
- *Scribtex* (<https://www.scribtex.com/>) is a place where you and collaborators can edit and compile L<sup>A</sup>T<sub>E</sub>X files. (With the free version you can have three projects with one collaborator per project.)
- *Online L<sup>A</sup>T<sub>E</sub>X Equation Editor* (<http://www.codecogs.com/latex/eqneditor.php>).

This page intentionally left blank.

## **L<sup>A</sup>T<sub>E</sub>X Package Conflicts**

[\[118\]](#) redirected me to [\[12\]](#) where I found a lot of L<sup>A</sup>T<sub>E</sub>X package conflicts. I think that this is one of the most up-to-date list of L<sup>A</sup>T<sub>E</sub>X package conflicts. It might help you.

This page intentionally left blank.

---

## Do you like my L<sup>A</sup>T<sub>E</sub>X Tips & Tricks?

It is often a good thing to look for advices, suggestions, and remarks about articles. My idea with these tips & tricks (be they coming from me, or from others, such as in the last articles) is to give you some pointers on how to achieve something which looks ‘simple’ in reality, but complicated (to the average user) in the L<sup>A</sup>T<sub>E</sub>X world. If you have any suggestion, remark, or any interesting idea, do not hesitate to contact me. A feedback is always welcome to improve these articles (I received only positive e-mails until here but you can also send negative remarks iff they are constructive).

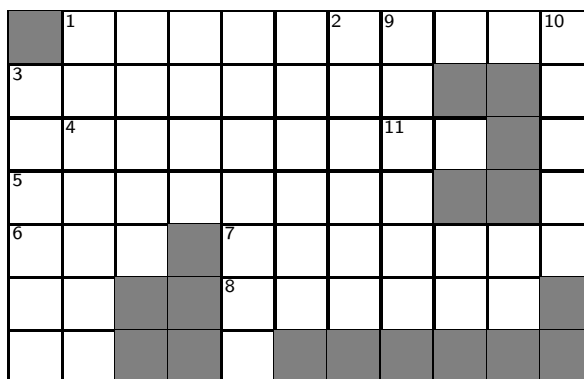
*There’s always room for improvement, you know, it’s the biggest room in the house.* – Louise Heath Leber

*Without continual growth and progress, such words as improvement, achievement, and success have no meaning.* – Benjamin Franklin

This page intentionally left blank.

## Test Crossword

Here is a test crossword. I managed to use different words from two subjects: T<sub>E</sub>X-related (mostly) and life-related. Note that the difficulty of this puzzle comes from the fact that you do not need to fill ‘all the available’ space with each word: just write each word, and if you succeed, every box will be filled at the end. *Proceed with fingers crossed!*



Here are some obscure clues.

**Across.** 1 Aligned in the middle 2 Not at left 3 Not with anybody 4 Analog to \ 5 Like an advocate 6 No clue! 7 What do you sometimes put in a figure? 8 No clue! 11 Ensure that something goes ‘out’ of the text

**Down.** 1 Not an engine 2 Roman 3 Has an extension mechanism 7 Insert something 9 Similar to slanted 10 Similar to subject

This crossword puzzle has been prepared using the `cwpuzzle` package. See [95] for more information. The solution is on next page.

Here is the solution of the puzzle.

	<sup>1</sup> C	E	N	T	E	<sup>2</sup> R	<sup>9</sup> I	G	H	<sup>10</sup> T
<sup>3</sup> L	O	N	E	L	Y	M	T			I
U	<sup>4</sup> N	E	W	L	I	N	<sup>11</sup> E	M		T
<sup>5</sup> A	T	T	O	R	N	E	Y			L
<sup>6</sup> T	E	X		<sup>7</sup> P	I	C	T	U	R	E
E	X			<sup>8</sup> U	S	E	B	O	X	
X	T			T						

---

## References

1. Universiteit Antwerpen. matlab2tikz, 2009. <http://win.ua.ac.be/~nschloe/content/matlab2tikz>.
2. Nelson H.F. Beebe. Outline and bitmap fonts compared , 2006. <http://www.math.utah.edu/~beebe/fonts/outline-vs-bitmap-fonts.html>.
3. Berkenbilt, Jay. (PDF specification message), 2010. <http://www.mail-archive.com/debian-user@lists.debian.org/msg570956.html>.
4. Leo Breebaart. How can you add empty pages to the end of a document?, 2010. <http://www.kronto.org/thesis/tips/empty-pages.html>.
5. Keith Briggs. Using truetype fonts with pdfL<sup>A</sup>T<sub>E</sub>X. <http://keithbriggs.info/ttf-pdflatex.html>.
6. Otfried Cheong. Using Truetype fonts and Unicode in PDFL<sup>A</sup>T<sub>E</sub>X. <http://tclab.kaist.ac.kr/ipe/pdftex.html>.
7. L<sup>A</sup>T<sub>E</sub>X Community (Forum). (Giving source URL in Figure environment), 2009. <http://www.latex-community.org/forum/viewtopic.php?f=44&t=5587>.
8. John D. Cook. Top four L<sup>A</sup>T<sub>E</sub>X mistakes, 2010. <http://www.johndcook.com/blog/2010/02/15/top-latex-mistakes>.
9. Richard D. Gill. Richard D. Gill’s home page – Product-integrals, 2010. <http://www.math.leidenuniv.nl/~gill/>.
10. Jon Dehdari. L<sup>A</sup>T<sub>E</sub>X hints, 2010. <http://www.ling.ohio-state.edu/~jonsafari/latex/>.
11. Glad Deschrijver. T<sub>E</sub>X Tricks, 2011. <http://users.ugent.be/~gdschrij/LaTeX/textricks.html>.
12. Freek Dijkstra. L<sup>A</sup>T<sub>E</sub>X Packages Conflicts, 2011. [http://www.macfreek.nl/mindmaster/LaTeX\\_package\\_conflicts](http://www.macfreek.nl/mindmaster/LaTeX_package_conflicts).
13. Michael Downes. Short Math Guide for L<sup>A</sup>T<sub>E</sub>X, 2002. <ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf>.
14. Existentialtype. Fonts in L<sup>A</sup>T<sub>E</sub>X, Part One: XeL<sup>A</sup>T<sub>E</sub>X, 2008. <http://existentialtype.net/2008/07/12/fonts-in-latex-part-two-pdftex-and-opentype/>.
15. Exomatik.net. L<sup>A</sup>T<sub>E</sub>X – Calculs, 2009. <http://www.exomatik.net/LaTeX/Calculs>.
16. Robin Fairbairns. New CTAN package: biblatex-ieee, 2011. <http://www.mail-archive.com/ctan-ann@dante.de/msg03710.html>.

17. Robin Fairbairns, Frank Mittelbach, and Werner Lemberg. L<sup>A</sup>T<sub>E</sub>X font encodings, 2006.
18. Robin Fairbairns, Philipp Stephani, and Luca Merciadri. Raster fonts questions, 2011. [http://groups.google.com/group/comp.text.tex/browse\\_thread/thread/3ef80548d126759a#](http://groups.google.com/group/comp.text.tex/browse_thread/thread/3ef80548d126759a#).
19. Fairbairns, Robin and Milde, Günter and Lehman, Philipp and Stephani, Philipp and Waßenhoven, Dominik and Merciadri, Luca. Encoding remarks (comp.text.tex discussion), 2010.
20. Daniel Flipo. Documentation sur le module frenchb de Babel, 2009. <http://daniel.flipo.free.fr/frenchb/frenchb2-doc.pdf>.
21. Peter Flynn. Typography. TUGboat, 28(2):172–173, 2007.
22. Peter Flynn. Formatting information; A beginner’s introduction to typesetting with L<sup>A</sup>T<sub>E</sub>X, 2009. <ftp://tug.ctan.org/tex-archive/info/beginlatex/beginlatex.pdf>.
23. Peter Flynn. (Personal communication.), 2011.
24. Peter Flynn. Formatting information: Chapter 7 – Textual tools, 2011. <http://latex.silmaril.ie/formattinginformation/chapter7.html#c184>.
25. Archlinux Forums. PDFL<sup>A</sup>T<sub>E</sub>X does not use Type 1 fonts. <https://bbs.archlinux.org/viewtopic.php?id=26485>.
26. PhilT<sub>E</sub>X Forums. So, is BibL<sup>A</sup>T<sub>E</sub>X the future?, 2011. <http://www.charlietanksley.net/philtex/forum/topic/so-is-biblatex-the-future>.
27. GL, Herbert Voß, and Luca Merciadri. Spacing in matrix with `\boxed` elements (comp.text.tex discussion), 2011.
28. Harvey J. Greenberg. A Simplified Introduction to L<sup>A</sup>T<sub>E</sub>X, 2006. <http://www.cs.usask.ca/~wew036/latex/simplified-intro/latex.pdf>.
29. Enrico Gregorio. Appunti di programmazione in L<sup>A</sup>T<sub>E</sub>X e T<sub>E</sub>X, 2009. Second edition (<http://profs.sci.univr.it/~gregorio/introtex.pdf>).
30. Gregorio, Enrico and Rudolph, Lee and Harper, John and Merciadri, Luca. Putting equation on a text line (comp.text.tex discussion), 2011.
31. Rene Grothmann. The most common Mistakes in L<sup>A</sup>T<sub>E</sub>X, 2011. <http://mga010.wordpress.com/the-most-common-mistakes-in-latex/>.
32. Gordon Grubert. True Type Fonts with L<sup>A</sup>T<sub>E</sub>X – Usage for L<sup>A</sup>T<sub>E</sub>X under Linux and MiK<sub>T</sub><sub>E</sub>X2.5. <http://fachschaft.physik.uni-greifswald.de/~stitch/ttf.html>.
33. HAPPYMUTANT.COM. A Quick & Dirty Guide to LaTeX – Making LaTeX Beamer Presentations, 2010. <http://happymutant.com/latex/misc/beamer.php>.
34. Robert Heller, Nicola Talbot, and Luca Merciadri. L<sup>A</sup>T<sub>E</sub>X, eBooks and reflowable PDFs questions, 2011. [http://groups.google.com/group/comp.text.tex/browse\\_thread/thread/45757e8435f04f2d#](http://groups.google.com/group/comp.text.tex/browse_thread/thread/45757e8435f04f2d#).
35. Heller, Martin and Maciel, Rui. Left brace on a subequations environment? (comp.text.tex discussion), 2010.
36. Gabor Herr. din.ist, 1991. <http://mirror.ctan.org/indexing/makeindex/ist/din.ist>.
37. Gabor Herr. icase.ist, 1991. <http://archive.cs.uu.nl/mirror/CTAN/indexing/makeindex/ist/icase.ist>.
38. Gabor Herr. latex.ist, 1991. <http://mirror.ctan.org/indexing/makeindex/ist/latex.ist>.
39. Gabor Herr. math.ist, 1991. <http://mirror.ctan.org/indexing/makeindex/ist/math.ist>.

40. Gabor Herr. `puncts.ist`, 1991. <http://mirror.ctan.org/indexing/makeindex/ist/puncts.ist>.
41. Gabor Herr. `tex.ist`, 1991. <http://mirror.ctan.org/indexing/makeindex/ist/tex.ist>.
42. Taco Hoekwater. OpenType fonts in Lua $\TeX$ . *TUGboat*, 29(1):34–35, 2008.
43. Hoekwater, Taco and Vieth, Ulrik. Surviving the  $\TeX$  font encoding mess; Understanding the world of  $\TeX$  fonts and mastering the basics of fontinst, 1999. <ftp://tug.ctan.org/tex-archive/fonts/utilities/fontinst/doc/talks/et99-font-tutorial.pdf>.
44. Klaus Hoppner. Truetype-Fonts in  $\LaTeX$ . <http://www.dalug.org/fileadmin/veranstaltungen/Slides/truetype.pdf>.
45. Micropress Inc. The OT1 font encoding. <http://www.micropress-inc.com/fonts/encoding/ot1.htm>.
46. Micropress Inc. The T1 font encoding. <http://www.micropress-inc.com/fonts/encoding/t1.htm>.
47. Rensselaer Polytechnic Insitute. Text Formatting with  $\LaTeX$ , 2007. <http://www.rpi.edu/dept/arc/docs/latex/latex-intro.pdf>.
48. The Odradek Institute. How to install OpenType fonts in  $\LaTeX$ . <http://members.fortunecity.com/odradek5/otf-LaTeX/index.html>.
49. Boguśław Jackowski. Latin Modern fonts at eleventh hour. <http://www.cstug.cz/aktivita/2005/lm-at11e.pdf>.
50. Christophe Jacquet. `Polices et pdflatex`, 2011. <http://www.jacquet80.eu/blog/post/2006/11/03/60-polices-et-pdflatex>.
51. Kalinowski, Eduardo and Luecking, Dan and Merciadri, Luca. ‘cases’ environment for a brace in the other sense? (comp.text.tex discussion), 2010.
52. Marie-Paule Kluth. Comment diviser une cellule par une diagonale?, 1997. <http://www.grappa.univ-lille3.fr/FAQ-LaTeX/7.13.html>.
53. Florian Knorn. M-code  $\LaTeX$  Package, 2009. <http://www.mathworks.com/matlabcentral/fileexchange/8015-m-code-latex-package>.
54. Torben Knudsen and Luca Merciadri. Including only frames in the bamer (comp.text.tex discussion), 2010.
55. Donald Ervin Knuth. *The  $\TeX$ book*. Addison-Wesley, 1986.
56. Korpela, Jukka. A tutorial on character code issues, 2009. <http://www.cs.tut.fi/~jkorpela/chars.html>.
57. G. Markus Kuhn. Effective scientific electronic publishing – Use Type 1 vector fonts for generating PDF files with  $\TeX$ , 2009. <http://www.cl.cam.ac.uk/~mgk25/publ-tips/#type1>.
58. Wikimedia Labs. *LaTeX/Tips and Tricks* – Wikimedia Labs, collection of open-content textbooks, 2010.
59. Leslie Lamport.  *$\LaTeX$ : A Document Preparation System*. Addison Wesley, 1994.
60. LaTeX-Community. Why type 3 fonts with `\usepackage[T1]{fontenc}`? <http://www.latex-community.org/forum/viewtopic.php?f=5&t=571>.
61. Pierre L’Ecuyer. Common mostly minor  $\LaTeX$  errors, 2009. <http://www.iro.umontreal.ca/~lecuyer/mcqm08/proceedings/latexery.pdf>.
62. Philipp Lehman. *The biblatex Package*, 2010. <http://www.ctan.org/tex-archive/macros/latex/exptl/biblatex/doc/biblatex.pdf>.
63. Philipp Lehman. *The biblatex Package*, 2011. <http://ftp.snt.utwente.nl/pub/software/tex/macros/latex/contrib/biblatex/doc/biblatex.pdf>.

64. David Lemon. Basic Type 1 Hinting. <http://typophile.com/files/hinting.pdf>.
65. Arno Linnemann. LaPrint Users Guide (LaPrint Version 3.16), 2004. <http://www.uni-kassel.de/fb16/rat/matlab/laprint/laprintdoc.ps>.
66. LiteratePrograms. Turing machine simulator ( $\text{\LaTeX}$ ), 2011. [http://en.literateprograms.org/Turing\\_machine\\_simulator\\_\(\LaTeX\)](http://en.literateprograms.org/Turing_machine_simulator_(\LaTeX)).
67. Locksley. How to include matlab source code in a  $\text{\LaTeX}$  document, 2009. <http://my.opera.com/locksley90/blog/2008/02/25/how-to-include-matlab-source-code-in-a-latex-document>.
68. Yannick Losser. Utilisation de polices TrueType avec  $\text{\LaTeX}$ . <http://pulsar68.org/latex/ttf/>.
69. LuaTeX. Lua $\text{\TeX}$  roadmap, 2011.
70. Lars Madsen. Various chapter styles for the memoir class, 2010. <http://ftp.ktug.or.kr/tex-archive/info/latex-samples/MemoirChapStyles/MemoirChapStyles.pdf>.
71. Lars Madsen, Heiko Oberdiek, and Luca Merciadri. Using the ref and label commands so that ref points to the label but with a user-given name (comp.text.tex discussion), 2010.
72. Madsen, Lars and Merciadri, Luca. How can I use `\boxed{}` in align environment? (comp.text.tex discussion), 2010.
73. MakeMoneyLog. Creating Your Own eBook - The Two Golden Rules When Writing eBooks For Profit, 2010. <http://www.makemoneylog.com/creating-your-own-ebook-the-two-golden-rules-when-writing-ebooks-for-profit.html>.
74. Lacheck manpage. Lacheck manpage, 2011. <http://www.man-online.org/page/1-lacheck>.
75. Gonzalo Arellano Medina. The background package, 2009. <http://mirrors.ctan.org/macros/latex/contrib/background/background.pdf>.
76. Medina Arellano, Gonzalo and Merciadri, Luca. (Space between address, etc., and the beginning of the text: lettre class - comp.text.tex | Google Groups, 2010. [http://groups.google.com/group/comp.text.tex/browse\\_thread/thread/048bbbf2d2537c39](http://groups.google.com/group/comp.text.tex/browse_thread/thread/048bbbf2d2537c39).
77. Luca Merciadri. P2P Implications on Web Surfing. P2P (Peer-to-Peer) technology comprises various ways to exchange information rapidly, each participant sharing a portion of his own resources. Anyway, despite of the numerous advantages of using P2P, a real problem is often encountered when using cheap internet connections: web surfing becomes so slow that it seems impossible to reach a web page, for the P2P's user. It is especially the case when using connections with a low upload speed. The problem has also an importance, even if it is minor, when using high-speed connections (VDSL, ...), as it is also a waste of capacity., 2009.
78. Luca Merciadri. The Ångström Manual, 2009. Manual of the Ångström Distribution.
79. Luca Merciadri. The dashundergaps package, 2009. Manual of the dashundergaps package.
80. Luca Merciadri. Some misunderstood or unknown tricks. TUGboat, 31:76–78, 2010.
81. Luca Merciadri. Some misunderstood or unknown tricks (II). TUGboat, 31(3):191–193, 2010.

82. Luca Merciadri. The bigints package, 2010. Manual of the bigints package.
83. Luca Merciadri. Merciadri packages: An overview. TUGboat, 32(2):206–210, 2011.
84. Luca Merciadri. Some misunderstood or unknown tricks (V) and encoding issues. TUGboat, in press.
85. Luca Merciadri, Marc Van Dongen, and Martin Münch. The turnthepage package, 2011. Manual of the turnthepage package.
86. Luca Merciadri, Marc Van Dongen, and Martin Münch. The turnthepage package, 2011. <http://mirror.ctan.org/macros/latex/contrib/turnthepage>.
87. Günter Milde and Luca Merciadri. Font encodings and PDFs, 2010.
88. Frank Mittelbach, David Carlisle, Chris Rowley, and Walter Schmidt. The fixltx2e and fix-cm packages, 2006. <http://www.tug.org/texmf-dist/doc/latex/base/fixltx2e.pdf>.
89. MobileRead. L<sup>A</sup>T<sub>E</sub>X template for Sony reader (PDF), 2007. <http://www.mobileread.com/forums/showthread.php?t=12872>.
90. MobileRead. Casanova, The Complete Memoirs – Comments, please!, 2009. <http://www.mobileread.com/forums/showthread.php?t=54190>.
91. MobileRead. L<sup>A</sup>T<sub>E</sub>X eBook Templates, 2009. <http://www.mobileread.com/forums/showthread.php?t=57861>.
92. MobileRead. PDF is not an eBook format, 2009. <http://www.mobileread.com/forums/showthread.php?t=22858>.
93. MrUnix.de. Gesamtanzahl Seiten, Abbildungen usw. - mrunix.de, 2010. <http://www.mrunix.de/forums/showthread.php?t=56716>.
94. Musa, Ahmed. The xwatermark Package, 2010. <http://www.ctan.org/tex-archive/macros/latex/contrib/xwatermark/xwatermark-guide.pdf>.
95. Gerd Neugebauer. A L<sup>A</sup>T<sub>E</sub>X Package for Typesetting Crossword Puzzles and More, 2009. Documentation date: 2009/09/13.
96. Heiko Oberdiek and Luca Merciadri. Footnotes in boxes (comp.text.tex discussion), 2010.
97. Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. The Not So Short Introduction to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>, 2003.
98. UK TeX (Archive of) Frequently Asked Questions. T<sub>E</sub>X Frequently Asked Questions – question label “addtoreset”, 2009. <http://www.tex.ac.uk/cgi-bin/texfaq2html?label=addtoreset>.
99. Martin J. Osborne. Some common L<sup>A</sup>T<sub>E</sub>X (T<sub>E</sub>X) errors, 2010. <http://www.economics.utoronto.ca/osborne/latex/LTXERR.HTM>.
100. John D. Owens. Installing OpenType Fonts in L<sup>A</sup>T<sub>E</sub>X with the LCD<sub>F</sub> Typetools. <http://www.ece.ucdavis.edu/~jowens/code/otfinst/>.
101. John D. Owens. The Installation and Use of OpenType Fonts in L<sup>A</sup>T<sub>E</sub>X. TUGboat, 27(2):112–118, 2006.
102. Scott Pakin. Comprehensive L<sup>A</sup>T<sub>E</sub>X Symbol list, 2008. <http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf>.
103. Scott Pakin. The Comprehensive L<sup>A</sup>T<sub>E</sub>X Symbol List, 2009.
104. T<sub>E</sub>X Frequently Asked Questions. Finding ‘8-bit’ Type 1 fonts. <http://www.tex.ac.uk/cgi-bin/texfaq2html?label=type1T1>.
105. Damir Rakityansky. Using TrueType fonts with T<sub>E</sub>X (L<sup>A</sup>T<sub>E</sub>X) and pdfT<sub>E</sub>X (pdfL<sup>A</sup>T<sub>E</sub>X). <http://www.radamir.com/tex/ttf-tex.htm>.

106. Massimo A. Redaelli. CircuitTikZ, 2009. <http://ctan.org/tex-archive/graphics/pgf/contrib/circuitikz/doc/latex/circuitikz/circuitikzmanual.pdf>.
107. Denis Roegel. The aeguill package, 2003. <http://archive.cs.uu.nl/mirror/CTAN/macros/latex/contrib/aeguill/guil-test1.pdf>.
108. Christopher Rose. L<sup>A</sup>T<sub>E</sub>X Gotchas – Common Pitfalls and Debugging, 2010. <http://www.eng.auburn.edu/~reevesj/Courses/ELEC6970-latex/gotch/LaTeXPres.pdf>.
109. Patrick Roux. Trente exemples mathématiques, 2008. <http://pagesperso-orange.fr/calque/latex/stage/exemplesmath.pdf>.
110. Diego Santa Cruz. High quality PDF output from L<sup>A</sup>T<sub>E</sub>X and T<sub>E</sub>X. <http://dsanta.users.ch/resources/type1.html>.
111. Michael Shell. How to Use the IEEEtran BibT<sub>E</sub>X Style, 2008. [http://ftp.snt.utwente.nl/pub/software/tex/macros/latex/contrib/IEEEtran/bibtex/IEEEtran\\_bst\\_HOWTO.pdf](http://ftp.snt.utwente.nl/pub/software/tex/macros/latex/contrib/IEEEtran/bibtex/IEEEtran_bst_HOWTO.pdf).
112. R. Slicht. The microtype package; An interface to the micro-typographic extensions of pdfT<sub>E</sub>X, 2009. <http://www.ctan.org/get/macros/latex/contrib/microtype/microtype.pdf>.
113. MacKichan Software. MacKichan Software - The Home of Sc. WorkPlace, Sc. Word, and Sc. Notebook, 2009. <http://www.mackichan.com/index.html?techtalk/574.htm~mainFrame>.
114. StackExchange. Latin Modern vs cm-super? <http://tex.stackexchange.com/questions/1390/latin-modern-vs-cm-super>.
115. StackExchange. Teletype `\textbackslash` in alltt environment. <http://tex.stackexchange.com/questions/16833/teletype-textbackslash-in-alltt-environment>.
116. StackExchange. Why should I use `\usepackage[T1]{fontenc}` ? <http://tex.stackexchange.com/questions/664/why-should-i-use-usepackagetifontenc/677#677>.
117. StackExchange. Average integral symbol, 2010. <http://tex.stackexchange.com/questions/759/average-integral-symbol>.
118. StackExchange. Big list – Packages that need to be included in a specific order, 2010. <http://tex.stackexchange.com/questions/3090/packages-that-need-to-be-included-in-a-specific-order>.
119. StackExchange. Big list – What useful web services are out there?, 2010. <http://tex.stackexchange.com/questions/3396/what-useful-web-services-are-out-there>.
120. StackExchange. Command to uppercase the first letter of each word in a sentence, 2010. <http://tex.stackexchange.com/questions/7992/command-to-uppercase-the-first-letter-of-each-word-in-a-sentence>.
121. StackExchange. Debugging – `\underfullrule`?, 2010. <http://tex.stackexchange.com/questions/4086/underfullrule>.
122. StackExchange. Differences between LuaT<sub>E</sub>X, ConT<sub>E</sub>Xt and XeT<sub>E</sub>X, 2010. <http://tex.stackexchange.com/questions/36/differences-between-luatex-context-and-xetex>.
123. StackExchange. Drawbacks of XeT<sub>E</sub>X/LuaT<sub>E</sub>X, 2010. <http://tex.stackexchange.com/questions/3094/drawbacks-of-xetex-luatex>.
124. StackExchange. How do I label different rows or columns of a matrix using braces?, 2010. <http://tex.stackexchange.com/questions/40/how-do-i-label-different-rows-or-columns-of-a-matrix-using-braces>.

125. StackExchange. Symmetric matrices, 2010. <http://tex.stackexchange.com/questions/3952/symmetric-matrices>.
126. StackExchange. What is the use of percent signs at the end of lines?, 2010. <http://tex.stackexchange.com/questions/7453/what-is-the-use-of-percent-signs-at-the-end-of-lines>.
127. StackExchange. What to do to switch to Bib $\text{\LaTeX}$ ?, 2010. <http://tex.stackexchange.com/questions/5091/what-to-do-to-switch-to-biblatex>.
128. StackExchange. Why do lower case mathcal letters show up as random symbols?, 2010. <http://tex.stackexchange.com/questions/8759/why-do-lower-case-mathcal-letters-show-up-as-random-symbols>.
129. StackExchange. Write date, time, and time zone, 2010. <http://tex.stackexchange.com/questions/8612/write-date-time-and-time-zone>.
130. StackExchange. Add asterisk after labels in enumerate, 2011. <http://tex.stackexchange.com/questions/21004/add-asterisk-after-labels-in-enumerate>.
131. StackExchange. Are there any (La) $\text{\TeX}$  Easter Eggs?, 2011. <http://tex.stackexchange.com/questions/9323/are-there-any-latex-easter-eggs>.
132. StackExchange. Are there widely used alternatives to Bib $\text{\TeX}$  that define their own file format?, 2011. <http://tex.stackexchange.com/questions/9951/are-there-widely-used-alternatives-to-bibtex-that-define-their-own-file-format>.
133. StackExchange. Automatically capitalize first letters of words in titles?, 2011. <http://tex.stackexchange.com/questions/23472/automatically-capitalize-first-letters-of-words-in-titles>.
134. StackExchange. Block quote with big quotation marks, 2011. <http://tex.stackexchange.com/questions/16964/block-quote-with-big-quotation-marks>.
135. StackExchange. Cases with square brackets, 2011. <http://tex.stackexchange.com/questions/12157/cases-with-square-brackets>.
136. StackExchange. Hiding the column separator in multicols, 2011. <http://tex.stackexchange.com/questions/19015/hiding-the-column-separator-in-multicols>.
137. StackExchange. How can I change the “References” to “Reference” in the `thebibliography` environment?, 2011. <http://tex.stackexchange.com/questions/17445/how-can-i-change-the-references-to-reference-in-the-thebibliography-environment>.
138. StackExchange. How should I draw a singly/double linked list?, 2011. <http://tex.stackexchange.com/questions/19286/how-should-i-draw-a-singly-double-linked-list>.
139. StackExchange. How to box every digit of an integer?, 2011. <http://tex.stackexchange.com/questions/22381/how-to-box-every-digit-of-an-integer>.
140. StackExchange. How to color the characters in  $\text{\LaTeX} 2_{\epsilon}$  logo?, 2011. <http://tex.stackexchange.com/questions/19916/how-to-color-the-characters-in-latex2e-logo>.
141. StackExchange. How to create my own math operator with limits?, 2011. <http://tex.stackexchange.com/questions/23432/how-to-create-my-own-math-operator-with-limits>.

142. StackExchange. How to have overlapping under-braces and over-braces, 2011. <http://tex.stackexchange.com/questions/12963/how-to-have-overlapping-under-braces-and-over-braces>.
143. StackExchange. How to typeset boldface ell?, 2011. <http://tex.stackexchange.com/questions/17199/how-to-typeset-boldface-ell>.
144. StackExchange. IEEE bibliography style for BibLaTeX (not BibTeX)?, 2011. <http://stackoverflow.com/questions/4595428/ieee-bibliography-style-for-biblatex-not-bibtex>.
145. StackExchange. Stacking Symbols – The `\bar` and `\overline` commands, 2011. <http://tex.stackexchange.com/questions/22100/the-bar-and-overline-commands>.
146. StackExchange. The line produced by `\not` looks bad on wider symbols, 2011. <http://tex.stackexchange.com/questions/23469/the-line-produced-by-not-looks-bad-on-wider-symbols>.
147. StackExchange. When should we use `\begin{center}` instead of `\centering`?, 2011. <http://tex.stackexchange.com/questions/23650/when-should-we-use-begincenter-instead-of-centering>.
148. StackExchange. Which tabular packages do which tasks and which packages conflict?, 2011. <http://tex.stackexchange.com/questions/12672/which-tabular-packages-do-which-tasks-and-which-packages-conflict>.
149. StackExchange. Why does L<sup>A</sup>T<sub>E</sub>X make a distinction between commands and environments?, 2011. <http://tex.stackexchange.com/questions/8373/why-does-latex-make-a-distinction-between-commands-and-environments>.
150. StackExchange. Why is L<sup>A</sup>T<sub>E</sub>X separated into many thousands of packages?, 2011. <http://tex.stackexchange.com/questions/15065/why-is-latex-separated-into-many-thousands-of-packages>.
151. Stackoverflow. How do I use TrueType fonts with L<sup>A</sup>T<sub>E</sub>X. <http://stackoverflow.com/questions/2525779/how-do-i-use-truetype-fonts-with-latex>.
152. Philipp Stephani and Luca Merciadri. Enumerate package: how to put two itemizations on the same line? (comp.text.tex discussion), 2010.
153. Till Tantau. The TikZ and PGF Packages, 2008. Manual for version 2.00; February 20.
154. Tantau, Till. TikZ, PGF, 2008. <http://www.ctan.org/tex-archive/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf>.
155. Christian Tellechea. L’extension ‘systeme’, 2011. <http://tug.ctan.org/pkg/systeme/>.
156. Bob Tennent and Luca Merciadri. Logic inferences in L<sup>A</sup>T<sub>E</sub>X (comp.text.tex discussion), 2011.
157. Texample. TikZ and PGF examples, 2010. <http://www.texample.net/tikz/examples/>.
158. Hàn Thê Thành, Sebastian Rahtz, and Hans Hagen. The pdfTeX manual, 2000. <http://tex.loria.fr/moteurs/pdftex-a.pdf>.
159. Manfred Thole. Apfelmännchen, 2009. <http://www.thole.org/manfred/apfel/>.
160. Sun Tong. L<sup>A</sup>T<sub>E</sub>X And TrueType Font. <http://xpt.sourceforge.net/techdocs/language/latex/latex33-LaTeXAndTrueTypeFont/>.
161. Edward R. Tufte. The Visual Display of Quantitative Information. 1983.
162. TUG. The L<sup>A</sup>T<sub>E</sub>X Font Catalogue. <http://www.tug.dk/FontCatalogue/>.

163. Tuteurs Enseignement. (Écrire une lettre avec LaTeX), 2010. <http://www.tuteurs.ens.fr/logiciels/latex/lettre.html>.
164. Jacobs University. L<sup>A</sup>T<sub>E</sub>X Poster - Teamwork at Jacobs University, 2009. <https://teamwork.jacobs-university.de:8443/confluence/display/CoPandBiG/LaTeX+Poster>.
165. Marc van Dongen. L<sup>A</sup>T<sub>E</sub>X and Friends. 2009. <http://www.cs.ucc.ie/~dongen/LaTeX-and-Friends.pdf>.
166. Marc Van Dongen and Luca Merciadri. Turn the Page, comp.text.tex, 2011.
167. Ulrik Vieth. Surviving the T<sub>E</sub>X font encoding mess. <http://www.scribd.com/doc/2184814/Surviving-the-TeX-font-encoding-mess>.
168. Paul Vojta. Using Cyrillic Symbols in L<sup>A</sup>T<sub>E</sub>X, 2005. <http://math.berkeley.edu/~vojta/tex/samp-1/sha.html>.
169. Vollmer, Jürgen. The draftcopy package, 2006. <http://www.ifi.uio.no/it/latex-links/draftcopy.pdf>.
170. Vladimir Volovich. CM-Super: Automatic creation of efficient Type 1 fonts from METAFONT fonts. TUGboat, 24(1):75–78, 2003.
171. Voß, Herbert and Gregorio, Enrico and Merciadri, Luca. dvi, ps, pdf in black and white: how to do it if I have colors? (comp.text.tex discussion), 2010.
172. ‘Night Walker’. Roman numerals in L<sup>A</sup>T<sub>E</sub>X, 2007. <http://nw360.blogspot.com/2007/09/roman-numerals-in-latex.html>.
173. Warp. L<sup>A</sup>T<sub>E</sub>X ASCII Mandelbrot, 2005. <http://warp.povusers.org/MandScripts/latex.html/>.
174. LyX Wiki. LyX wiki | BibL<sup>A</sup>T<sub>E</sub>X, 2011. <http://wiki.lyx.org/BibTeX/Biblatex>.
175. Wikibooks. L<sup>A</sup>T<sub>E</sub>X, 2009.
176. Wikipedia. LaTeX/Internationalization – Wikibooks, collection of open-content textbooks, 2010. <http://en.wikibooks.org/wiki/LaTeX/Internationalization>.
177. Wikipedia. ConT<sub>E</sub>Xt, 2011.
178. Wikipedia. ConT<sub>E</sub>Xt (French), 2011.
179. Wikipedia. L<sup>A</sup>T<sub>E</sub>X Internationalization – Output encoding, 2011. [http://en.wikibooks.org/wiki/LaTeX/Internationalization#Output\\_encoding](http://en.wikibooks.org/wiki/LaTeX/Internationalization#Output_encoding).
180. Wikipedia. Interrobang, 2011.
181. Wikipedia. Irony punctuation, 2011.
182. Wikipedia. LuaT<sub>E</sub>X, 2011.
183. Wikipedia. Sha (III), 2011. <http://en.wikipedia.org/wiki/Shas>.
184. Wikipedia. Shuffle product, 2011. [http://en.wikipedia.org/wiki/Shuffle\\_product](http://en.wikipedia.org/wiki/Shuffle_product).
185. Wikipedia. L<sup>A</sup>T<sub>E</sub>X/Tables – Wikibooks, 2011. <http://en.wikibooks.org/wiki/LaTeX/Tables>.
186. Wikipedia. TeT<sub>E</sub>X, 2011.
187. Wikipedia. Turing completeness, 2011.
188. Wikipedia. XeT<sub>E</sub>X, 2011.
189. Wilson, Peter. Some Examples of Title Pages, 2009.
190. Andrew T. Young. Controlling L<sup>A</sup>T<sub>E</sub>X floats, 2010. <http://mintaka.sdsu.edu/GF/bibliog/latex/floats.html>.

This page intentionally left blank.

---

## Index

- [%, 99](#)
- [III, 116](#)
- [AutoMath, 127](#)
- [background, 146](#)
- [beamerarticle, 91](#)
- [BibLaTeX, 169](#)
- [BibTeX, 169](#)
- [Biber, 169](#)
- [bigint, 63, 66](#)
- [bigints, 66](#)
- [bigintss, 66](#)
- [bigintsss, 66](#)
- [bigintssss, 66](#)
- [Block quote, 20](#)
- [Bmatrix, 68](#)
- [bmatrix, 68](#)
- [Braces, 69](#)
- [Calc2LaTeX, 45](#)
- [Captions, 46](#)
- [center, 51](#)
- [centering, 51](#)
- [Circuits, 28](#)
- [cleardoublepage, 97](#)
- [cm-super, 177](#)
- [Column separator, 113](#)
- [Command, 114](#)
- [Counter, 11](#)
  - [chapter, 11](#)
  - [enumiii, 11](#)
  - [enumii, 11](#)
  - [enumiv, 11](#)
  - [enumi, 11](#)
  - [equation, 11](#)
  - [figure, 11](#)
  - [footnote, 11](#)
  - [mpfootnote, 11](#)
  - [page, 11](#)
  - [paragraph, 11](#)
  - [part, 11](#)
  - [section, 11](#)
  - [subparagraph, 11](#)
  - [subsection, 11](#)
  - [subsubsection, 11](#)
  - [table, 11](#)
- [CP1252, 150](#)
- [Crops, 97](#)
- [Dash integrals, 56](#)
- [dashuline, 57](#)
- [deutsch, 89](#)
- [Diagonal, 30](#)
- [dimen, 41](#)
- [Display of date and time, 78](#)
- [dotuline, 57](#)
- [draftcopy, 146](#)
- [eBooks, 193](#)
- [english, 89](#)
- [Enumerate, 52](#)
- [Environment, 114](#)
- [Fncychap, 41](#)
- [fontenc, 175](#)
- [français, 89](#)
- [Front cover, 23](#)

- `inputenc`, 149, 175
- Interrobang, 103
- Ironicon, see Percontation point
- ISO-8859-1, 150
- 
- `lacheck`, 204
- `laprint`, 43
- Learning curve, 201
- `lettre`, 100
- Linebreak, 7
- Linked list, 24
- `lmodern`, 177
- Logic gates, 28
- `LuaTeX`, 150
- 
- `mag`, 47
- `makeatletter`, 71
- `makeatother`, 71
- `marginpar`, 46
- Margins, 46
- `matfig2pgf`, 44
- `mathcal`, 76
- MATLAB, 43
- `matlab2tikz`, 45
- `mcode`, 102
- `METAFONT`, 175
- Minipage, 49
- Minitoc, 74
- Monochrome, 48
- 
- `nederlands`, 89
- 
- OpenType (font), 178
- OT1, 154, 176
- `overline`, 25
- 
- Percontation point, 103
- Plagiarism, 157
- 
- `pmatrix`, 68
- Product integrals, 75
- 
- QED square, 95
- Quotation mark, 18
- 
- `SageTeX`, 45
- `scalebox`, 51
- Sha, see III
- `short`, 89
- Slash, 22
- Slave counter, 12
- Spreadsheet, 45
- Square brackets, 27
- Square roots, 28
- Stacks, 77
- 
- T1, 177
- `texttt`, 83
- Tikz, 33
- Titlepage, 19
- TrueType (font), 178
- Turing (-completeness), 163
- `turnthepage`, 88
- Type
  - 1 (font), 175
  - 3 (font), 175
- 
- `utf8`, 149
- `utf8x`, 149
- 
- `Vmatrix`, 68
- `vmatrix`, 68
- 
- watermark, 145
- 
- `XeTeX`, 150
- `xwatermark`, 146