Numero 28
Ottobre
2019

ArsTEXnica

Rivista italiana di TEX e LATEX

GuIT

http://www.guitex.org/arstexnica/

ArsTeXnica

GuIT – Gruppo Utilizzatori Italiani di TeX

ArsTeXnica è la pubblicazione ufficiale del GuIT

ArsTeXnica è la prima rivista italiana dedicata a TeX, a LaTeX ed alla tipografia digitale. Lo scopo che la rivista si prefigge è quello di diventare uno dei principali canali italiani di diffusione di informazioni e conoscenze sul programma ideato quasi trent'anni fa da Donald Knuth.

Le uscite avranno, almeno inizialmente, cadenza semestrale e verranno pubblicate nei mesi di Aprile e Ottobre. In particolare, la seconda uscita dell'anno conterrà gli Atti del Convegno Annuale del GuIT, che si tiene in quel periodo.

La rivista è aperta al contributo di tutti coloro che vogliano partecipare con un proprio articolo. Questo dovrà essere inviato alla redazione di ArsTeXnica, per essere sottoposto alla valutazione di recensori. È necessario che gli autori utilizzino la classe di documento ufficiale della rivista; l'autore troverà raccomandazioni e istruzioni più dettagliate all'interno del file di esempio (`.tex`). Tutto il materiale è reperibile all'indirizzo web della rivista.

Gli articoli potranno trattare di qualsiasi argomento inerente al mondo di TeX e LaTeX e non dovranno necessariamente essere indirizzati ad un pubblico esperto. In particolare tutorials, rassegne e analisi comparate di pacchetti di uso comune, studi di applicazioni reali, saranno bene accetti, così come articoli riguardanti l'interazione con altre tecnologie correlate.

Di volta in volta verrà fissato, e reso pubblico sulla pagina web, un termine di scadenza per la presentazione degli articoli da pubblicare nel numero in preparazione della rivista. Tuttavia gli articoli potranno essere inviati in qualsiasi momento e troveranno collocazione, eventualmente, nei numeri seguenti.

Chiunque, poi, volesse collaborare con la rivista a qualsiasi titolo (recensore, revisore di bozze, grafico, etc.) può contattare la redazione all'indirizzo:

arstexnica@guitex.org.

Nota sul Copyright

Associarsi a GuIT

Fornire il tuo contributo a quest'iniziativa come membro, e non solo come semplice utente, è un presupposto fondamentale per aiutare la diffusione di TeX e LaTeX anche nel nostro paese. L'adesione al Gruppo prevede una quota di iscrizione annuale diversificata: 30,00 € soci ordinari, 20,00 (12,00) € studenti (junior), 75,00 € Enti e Istituzioni.

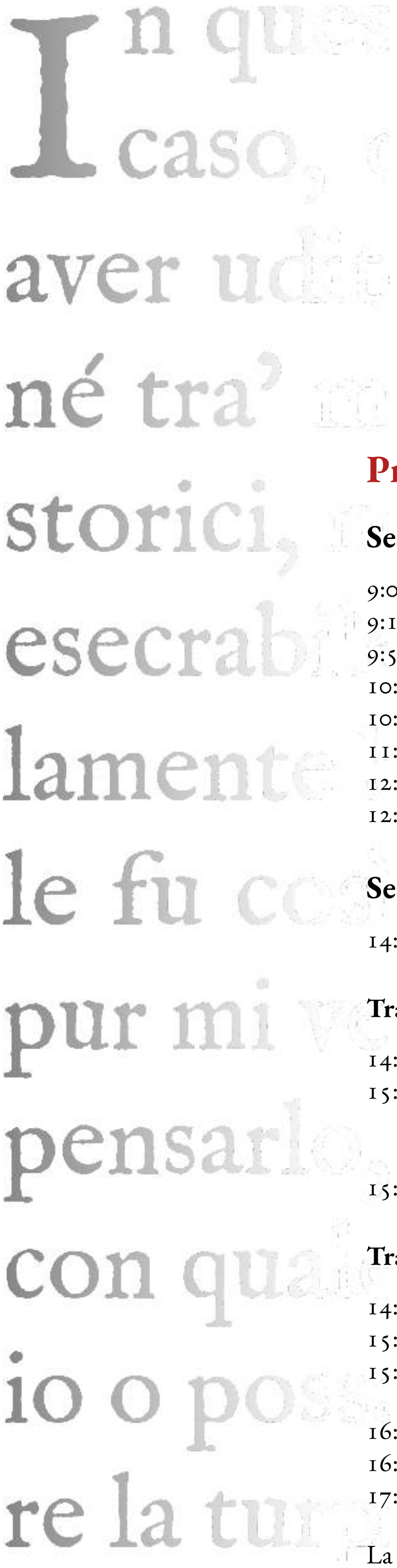
Indirizzi

Gruppo Utilizzatori Italiani di TeX
c/o Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Industriale
Via Claudio 21
80125 Napoli – Italia
`http://www.guitex.org`
guit@guitex.org

Redazione ArsTeXnica:
`http://www.guitex.org/arstexnica/`
arstexnica@guitex.org

Codice ISSN 1828-2369

15 Ottobre 2019



GuIT meeting 2019

SEDICESIMO CONVEGNO NAZIONALE SU TEX, LATEX E TIPOGRAFIA DIGITALE

26 ottobre 2019

Politecnico di Torino

Sala Consiglio di Facoltà,
Corso Duca degli Abruzzi 24, Torino

Programma del convegno

Sessione mattutina (corso introduttivo a LaTeX)

9:00 Benvenuto.
9:10 *Introduction to LaTeX and to some of its tools*. G. Pignalberi and M. Dominici.
9:50 *TeX, LaTeX and math*. E. Gregorio.
10:30 — Pausa caffè (15 min).
10:45 *Bibliographies, LaTeX and friends*. G. Milanese.
11:15 *Graphics for LaTeX users*. A. De Marco.
12:00 *Presentations with Beamer*. G. Messineo and S. Vassallo.
12:30 *The* Toptesi *package. Typesetting a PhD thesis with LaTeX*. C. Beccari.

Sessione pomeridiana

14:30–16:00 Help Desk a cura di volontari del Gruppo Utilizzatori Italiani di TeX.

Traccia A

14:30 *Creating accessible pdfs with LaTeX*. U. Fischer.
15:00 *Axessibility 2.0: creating tagged PDF documents with accessible formulae*. D. Ahmetovic, T. Armano, C. Bernareggi, A. Capietto, S. Coriasco, B. Doubrov, A. Kozlovskiy and N. Murru.
15:30 *Uno script bash di ausilio alla redazione di manoscritti*. G. Pignalberi.

Traccia B

14:30 *A Direct Bibliography Style for ArsTeXnica*. J.-M. Hufflen.
15:00 *Smartdiagram: The Package and Its Journey*. C. Fiandrino.
15:30 *Metamorfosi dei tipi sublacensi*. C. Vincoletto.

16:00 — Pausa caffè (15 min).
16:15 Riunione annuale del gruppo.
17:30 Chiusura dei lavori.

La partecipazione è libera e gratuita, previa registrazione entro il 20 ottobre.
Registrazione online: `www.guitex.org/home/meeting`

Gruppo Utilizzatori Italiani di TeX
GuIT

www.guitex.org

POLITECNICO DI TORINO

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 28, Ottobre 2019

Gruppo Utilizzatori Italiani di TEX

Editoriale

Claudio Beccari

Questo numero di ArsTeXnica è decisamente particolare. Il Meeting nel quale vengono presentati questi lavori è ospitato per la prima volta al Politecnico di Torino, l'Ateneo dove ho lavorato per 45 anni prima di andare in pensione e a cui sono evidentemente affezionato. La Scuola di Dottorato di questo Ateneo non mi ha mai visto come docente, ma per la Scuola avevo già da tempo predisposto un modello e una classe per la composizione delle tesi dottorali. In occasione di questo Meeting la Scuola ha chiesto al GuIT di tenere una specie di corso di approfondimento per i dottorandi.

Con il supporto del Politecnico e della sua Scuola di Dottorato la prima metà di questo Meeting è dedicata ad un ciclo di "lezioni" da presentare ai dottorandi partendo dall'inizio per finire con l'uso avanzato della classe per la composizione delle tesi di dottorato, che raccoglie un po' tutti gli aspetti di un uso abbastanza approfondito di LaTeX.

Ringrazio quindi il Politecnico di Torino, nelle persone del suo Rettore Guido Saracco e del Vicerettore addetto alla didattica Sebastiano Foti, e la sua Scuola di Dottorato, nella persona del Direttore Stefano Grivet Talocia, per il supporto fornito al GuIT.

In questa occasione la CLUT, la Cooperativa Libraria Universitaria di Torino, che ha sede presso il Politecnico, ha fornito i suoi servizi per la stampa del volume che contiene sia ArsTeXnica 27 sia ArsTeXnica 28; quest'ultimo numero raccoglie sia le lezioni in inglese per la Scuola di dottorato, sia agli articoli normali presentati nella seconda metà del Meeting. La CLUT, che ringrazio nella persona del suo Direttore Michele Ruffino, è una delle poche case editrici che richiede ai suoi autori di consegnare i testi da pubblicare composti preferibilmente con LaTeX; da più di venti anni essa fornisce ai suoi autori la propria classe `clut.cls` con tutta la documentazione del caso, in modo che essi non debbano preoccuparsi d'altro che scrivere il loro testo.

Questo è anche l'ultimo numero che curo come Direttore di questa rivista; compiendo gli ottanta anni ritengo che sia giunta l'ora di lasciare il posto ad un nuovo Direttore più giovane di me; egli dovrebbe venire nominato dal Consiglio Direttivo del GuIT durante questo stesso Meeting o nei giorni successivi. Ringrazio quindi il GuIT per avere avuto fiducia in me affidandomi l'onore e l'onere di gestire la pubblicazione degli ultimi dodici numeri della rivista.

Ringrazio inoltre tutti gli autori, e tutti coloro che hanno collaborato per la realizzazione non solo di questo numero di ArsTeXnica, ma anche di tutti i numeri precedenti: la Redazione, il Consiglio Scientifico, e i numerosi revisori editoriali.

Comincio quindi a presentare le lezioni per i dottorandi della Scuola di Dottorato; esse sono esposte con la stessa sequenza temporale con cui sono presentate ai dottorandi.

Gianluca Pignalberi e Massimiliano Dominici presentano un'introduzione iniziale a LaTeX dove mostrano le differenze che si ottengono componendo con LaTeX rispetto a quello che si può ottenere con i vari word processor, che sono tanto di moda ma che non si avvicinano al risultato ottenibile con LaTeX nemmeno usandoli in modo professionale. L'articolo è accompagnato da molte figure di libri composti dagli autori dalle quali si possono rilevare le numerose possibilità di composizione offerte da questo sistema.

Per gli allievi della scuola di dottorato che si occupano di discipline tecnico/scientifiche la composizione della matematica è di fondamentale importanza. Farlo con LaTeX non è difficile, casomai è difficile il linguaggio della matematica. Enrico Gregorio fornisce i fondamenti della composizione mediante LaTeX, ma insiste molto sulle cose da *non* fare. Certi errori sono comunissimi e per comporre bene bisogna conoscerli per evitarli. Accenna alle norme ISO relative alla matematica delle grandezze e cita le norme del Sistema Internazionale senza scendere in troppi dettagli; in fondo la documentazione è abbondante, e gli allievi sono abituati a leggere testi contenenti matematica delle grandezze; con questa lezione riescono anche a individuare gli errori commessi da professionisti in queste discipline.

La bibliografia nelle tesi di laurea e di dottorato è una parte essenziale, qualunque sia la disciplina su cui la tesi verte. Guido Milanese è un letterato e in tale qualità sa bene come ricavare le informazioni necessarie e complementari di numerosi archivi bibliografici a disposizione; conosce bene i modi di citare le opere e di presentare nell'elenco bibliografico tutte le informazioni necessarie per la corretta informazione per i lettori delle opere stampate, comprese le tesi, che spesso sono il punto di partenza per altri laureandi o dottorandi per proseguire le ricerche su determinati argomenti.

Va da sé che ogni tesi nelle discipline tecnico/scientifiche è corredata da immagini di vario genere; il disegno programmato permette di comporre disegni molto più professionali di quanto si può ottenere con una semplice interfaccia grafica che si affidi solo al mouse. Agostino De Marco ha una vasta esperienza in merito, sia per quello che riguarda i programmi di disegno facenti parte del sistema TEX, sia per un certo numero di programmi liberi o commerciali che però hanno la possibilità di esportare i loro risultati in formato compatibile con il sistema TEX. Le informazioni e gli esempi mostrati sono molto validi e permettono di guidare l'utente verso le soluzioni più idonee.

Grazia Messineo e Salvatore Vassallo hanno una lunga esperienza alle spalle con la divulgazione di LATEX; essi quindi hanno sviluppato un'ottima esperienza con le presentazioni da proiettare in aule o sale di conferenze; ovviamente le loro proiezioni sono realizzate con i mezzi stessi di LATEX, fra i quali spicca la classe beamer. Anche i laureandi e i dottorandi ad un certo punto del loro percorso di studi devono discutere la loro tesi ed hanno bisogno di un mezzo per proiettare la loro presentazione che non abbia i limiti consueti dei programmi WYSIWYG, generalmente ottimi e pieni di effetti speciali, ma lacunosi per quel che riguarda la matematica. La loro esperienza didattica traspare in questa lezione dedicata ai dottorandi.

Claudio Beccari mette insieme tutte le informazioni precedenti per descrivere la composizione delle tesi di vario tipo che debbono essere composte dagli studenti alla fine di ogni loro ciclo di studi; egli dedica una particolare attenzione alle tesi dottorali da svolgere presso la Scuola di Dottorato del Politecnico di Torino che preferisce che siano composte con LATEX con la prerogativa di essere conformi alle norme ISO per l'archiviabilità a lungo termine; le norme ISO sono molto stringenti e non sono facili da rispettare per una quantità di motivi che vengono illustrati e discussi. In conclusione Beccari suggerisce di servirsi del programma di composizione LuaLATEX: inoltre raccomanda di usare il pacchetto TOPtesi che contiene anche un modello (*template*) già configurato sia per rispettare i requisiti richiesti dalla Scuola di Dottorato sia quelli richiesti dalle norme ISO per l'archiviabilità.

Passo ora a presentare i lavori presentati da utenti di LATEX in ordine alfabetico degli autori.

Presso l'Università di Torino un gruppo di ricerca, confluente nel Laboratorio Polin coordinato dalla professoressa Anna Capietto, si occupa da anni del problema di rendere accessibili i testi che contengono della matematica agli studenti affetti da disabilità visive di vario genere compresa la cecità totale. Questo gruppo ha già presentato diversi suoi lavori sia nei Meeting del GuIt, sia nelle Conferenze internazionali del TUG (TEX Users Group) in merito al progredire dei loro lavori. Questo articolo descrive lo stato dell'arte alla luce degli ultimi risultati ottenuti, che permettono ai disabili visivi di ascoltare la lettura ad alta voce ottenibile con gli specifici programmi dei vari sistemi operativi quando i file PDF sono stati composti con le estensioni create dal gruppo di ricerca per la lettura anche della matematica composta con LATEX.

Claudio Fiandrino negli anni ha sviluppato una competenza particolare per gestire la grafica mediante le funzionalità del sistema TEX costituite dai pacchetti Ti*k*Z e pgfplots. In questo articolo egli presenta un modulo di libreria Ti*k*Z adatto per comporre certi diagrammi di flusso che richiedono configurazioni speciali per connettere fra loro diversi blocchi secondo alcuni schemi predefiniti. La particolarità di questo modulo è che bastano poche parole chiave per descrivere il tipo di schema e il modulo fa tutto da solo. Gli esempi riportati sono decisamente interessanti e dimostrano con chiarezza le funzionalità del modulo.

Ulrike Fischer è il membro del LATEX Team che si occupa della creazione dei file *tagged PDF* e dell'accessibilità. Il suo lavoro prevede diversi anni di studi e prove di modifiche dei file che costituiscono il nucleo di LATEX e delle principali classi. La proprietà *tagged* dei file PDF allo stato attuale non è realizzata con i motori di composizione pdftex, e xetex, ma è meno difficile da implementare se si usa come motore luatex. Questa proprietà rende possibile l'accessibilità ai documenti PDF sia ai disabili che necessitano di dispositivi di *screen reading*, sia per altre funzioni che richiedono di accedere a certi dati contenuti dentro il file. Il breve e interessantissimo articolo contenuto in questo numero della rivista va completato con l'esposizione orale e con un altro articolo più approfondito che l'autrice ci ha promesso per il prossimo numero di ArsTEXnica.

Jean-Michel Hufflen si occupa da molto tempo delle bibliografie composte usando il programma di estrazione MLBibTEX (Multi Language BibTeX) insieme a certi pacchetti di stile tipografico adatti allo scopo. In questo articolo egli presenta un adattamento delle sue procedure per comporre le bibliografie di questa rivista ArsTEXnica, in modo da superare i limiti dell'attuale procedura che si affida al tradizionale programma BibTEX accompagnato da uno specifico pacchetto di stile bibliografico arstexnica.bst.

Gianluca Pignalberi da tempo si occupa dell'impaginazione con LATEX di testi da pubblicare presso case editrici, che nella maggior parte dei casi ricevono i testi da pubblicare mediante file composti con vari word processor, e quasi sempre non strutturati mediante gli appositi stili di quei programmi. Il lavoro necessario per strutturare questi file convertendoli in file `.tex` adatti per la composizione mediante i tre programmi principali di composizione basati sul linguaggio LATEX richiede un lungo lavoro monotono e specialmente soggetto a sviste ed errori, che si può evitare mediante script di vario genere (per macchine UNIX o Windows; rispettivamente procedure bash o bat) che permettono di eseguire le necessarie correzioni, o, almeno, di segnalare punti dei testi sorgente in cui il redattore debba guardare con particolare attenzione. È chiaro che per realizzare questi script e/o per estenderli ad altri casi è necessaria la conoscenza delle loro sintassi unita a quella del linguaggio LaTeX.

Claudio Vincoletto da tempo si occupa dei caratteri da stampa, moderni e antichi, e del loro studio. In questo articolo descrive gli studi che lui ha fatto sul font storico Subiaco, inizialmente realizzato dagli incisori tedeschi Sweynheym e Pannartz, già allievi di Fust e Schöffer, a loro volta allievi e soci di Gutenberg, trasferitisi prima a Subiaco poi a Roma. Questo font imita la scrittura manuale usata dagli amanuensi nel monastero di Subiaco. Nonostante l'origine tedesca degli incisori e del loro apprendistato in Germania, il font da essi creato è ispirato alla calligrafia umanistica italiana e rappresenta una anticipazione rispetto ai caratteri tondi e corsivi sviluppati successivamente nel Nord Italia. Vincoletto ne ha ricreato la versione elettronica di tipo vettoriale, usando principalmente il programma METAFONT, e poi uno dei vari programmi di conversione per ottenere dal file sorgente contenente il codice METAFONT direttamente le forme vettoriali comuni ai font Type 1 e OpenType. L'analisi storica, estetica e programmatica di questo articolo è molto interessante ed estende in modo insolito le conoscenze degli utenti di LaTeX.

▷ Claudio Beccari
Professore emerito
Politecnico di Torino
`claudio dot beccari at gmail dot com`

Introduction to LaTeX and to some of its tools

Gianluca Pignalberi, Massimiliano Dominici

Abstract

Writing has a long history. Shorter is the history of typesetting and even shorter is the history of digital typography. Nevertheless, the latter gained an unprecedented importance because of its capability to speed up the process of feeding human being with well-composed information.

Our lessons, of which this is the number zero, are focused on a digital typesetting system that has come to light in the late 70s of 1900. It was intended to typeset scientific books; it is used to typeset nearly everything. It is TeX.

In this short course we will give an overview on how TeX and its most famous macro package LaTeX helps engineers, scientists and professionals to compose their documents, being them books, papers, reports, presentations, posters.

Sommario

La scrittura ha una lunga storia. Più breve è la storia della composizione tipografica e ancor più breve è quella della tipografia digitale. Ciò nonostante, quest'ultima ha guadagnato un'importanza senza precedenti per la sua capacità di velocizzare il processo di rifornire l'essere umano di informazione ben composta.

Le nostre lezioni, di cui questa è la numero zero, sono incentrate su un sistema di composizione digitale venuto alla luce nei tardi anni '70 del 1900. Questo era pensato per comporre libri scientifici; è usato per comporre quasi tutto. È TeX.

In questo breve corso daremo una panoramica di come TeX e il suo più famoso pacchetto di macro LaTeX aiuta gli ingegneri, gli scienziati e i professionisti a comporre i loro documenti, siano essi libri, articoli, relazioni, presentazioni, poster.

Part I: Digital Typography and Not

1 Typesetting Systems *vs* Word Processors

Computers have often been (and currently are) used as typewriters, i.e., to edit text. Text editing has evolved: from the simple words juxtaposition with no hyphenation, monolingual spell check, monospaced font and fixed spacing (between words and lines) and dimensions (for the document) to character kerning, spell and grammar check in different languages, fancy OpenType fonts with contextual shapes and better page and document setup.

Text processing is wonderfully performed by word processors (WPs from now on): Word, Writer and, back in time, DecWrite, WordPerfect, LetterPerfect, WordStar... But, while WordStar and LetterPerfect were not that fancy (they existed when printers were light years far from the Apple LaserWriter), the others listed after Word and Writer were closer to modern WPs. While ancient WPs just allowed text arrangement on the page (no external elements were allowed; no font selection; no fanciness) and spell check, modern WPs are capable of much, much more. They have partially invaded the world of typesetting systems: some minor publishing houses use WPs to create their camera-ready books and journals and we personally set up a LaTeX class for such a publishing house that used to typeset an academic journal in Word. Even a Mid-Pharma company used Word to produce its official reports... until someone discovered that Word was not able to include something like 800 PDF automatically-generated tables into the same report and tried to switch to LaTeX through LyX. Since WPs and typesetting systems have different targets, WPs are not yet as sharp and versatile as typesetting systems are and typesetting systems do not see text as their "core business", i.e., they are not intended as "click-and-type" programs.

Nobody less than insane would pretend to compare the performances of such different tools. WPs are programs mainly intended to process text and to let unskilled users produce reports and other documents with a decent look. Typesetting systems (once named DTP after DeskTop Publishing) are powerful and highly specialized tools to professionally produce newspapers, magazines, books, journals, fliers, banners, you name it. They just process text as one of the zillions tasks they do but have to typeset it the best way as possible. LaTeX *is* a typesetting system and so it is its typesetting engine: TeX. As we will see in the next section, being LaTeX a command line system, it even has to rely on an external text editor. The aforementioned target difference and the LaTeX lack of a built-in text editor turns useless those endless discussions about "Is it LaTeX better than Word?", as in Oetiker *et al.* (2018, pp. 3–4) (that shows pros and cons), Knauff and Nejasmic (2014) and Blanco (2015) (that advise Word) or Bloch (2017) (that explains why LaTeX should be better). It is pointless to compare two programs that perform different, though partially superimposed, tasks. It is pointless to measure how fast users input text and tables when the hardware is not the same and you do not specify who had the autocorrect activated or not: this is a word processors tool—not typesetting systems'—that might increase the speed performance of users. It is pointless to measure how sharp users input text and tables and not to measure how close to the original is the final document look (but Knauff and Nejasmic (2014) mentions that).

WPs care about the fact that every line in a page is good and well hyphenated, regardless of the page quality; typesetting systems not only care that a line is more than good and correctly typeset and hyphenated but should care of the page quality too. TEX only issues a page when the page is typographically the best possible according to TEX's internal rules. At last, Word does not care of documents back-compatibility, unlike LATEX. More on that topic in section 4.

2 Interactive and Non-Interactive Typesetting Systems

Despite the majority of users just know interactive[1] programs, there are still many programs that are not interactive. Typesetting systems are no exception and probably the most representative non-interactive programs and troff and TEX. Interactive (and visual) typesetting systems are QuarkXPress, Adobe InDesign, Microsoft Publisher, Scribus, and the very ancient (and dismissed) Ventura Publisher and Aldus PageMaker.

The main difference between interactive and non-interactive typesetting systems is that an interactive one shows you in real time the result of your actions and your actions are usually dragging and dropping boxes in a visual interface, while a non-interactive program accepts whatever action you want but shows you a result only when you instruct it to show. So, if you delete a sentence in an InDesign text, you immediately see the modified text; if you do the same with TEX, you will only see the result after compiling the new text and opening the resulting DVI or PDF.

Just to roughly cut the users set, graphic designers consider more productive using interactive programs; programmers think otherwise and it seems to us that programmers are closer to old linotypers, who had to rely on their experience to produce text lines that where not too empty or too full before seeing them cast into lead.

We are now entering the world of an old, yet way too powerful typesetting system: TEX, and its almost universally used "hi-level access gate": LATEX.

3 TEX As a Non-Interactive Typesetting System and a Programming Language

TEX is a typesetting system designed and programmed by Donald Knuth at the Stanford University. Its first release came to light in 1978. In Knuth (1999, chap. 1) the author explains why he decided to write such a program: the first volumes of his masterpiece *The Art of Computer Programming*, first typeset with Monotype, needed to be updated but that technology had been dismissed in the USA. The available technology was unable to get at least a similar result so he decided that a program able to typeset books and a program able to generate the needed fonts had to be written: they where TEX and METAFONT.

TEX is a non-interactive typesetting system, so the users have to instruct it—program it—on how to output the desired document. Once the program is ready, TEX compiles it and—hopefully—outputs the document (in DVI format).

TEX programming language provides the users about 900 commands, tests and so on. It is straightforward to realize that TEX is extremely powerful, yet not much user-friendly in the way we currently intend that friendliness.

The DVI documents needed one more step to be ready for a printing service: a DVI→PostScript conversion had to be performed. The program `dvips` accomplished that task.

Years later pdfTEX started outputting PDF documents.

These programs have been partially superseded by XƎTEX and LuaTEX: both support TTF/OTF fonts and LuaLATEX adds to TEX a powerful yet simple programming language: Lua.

As you may figure out, TEX is not a program that can be installed by itself, without any companion program or file. Indeed it comes with packages known as *distributions*. Despite several distributions have been available for different operating systems, it is now common to see that three distributions polarized users: MikTEX (`https://miktex.org/`) on Windows systems, TEX Live (`https://www.tug.org/texlive/`) on Linux systems and MacTEX (`https://www.tug.org/mactex/`) on Apple computers. While you can refer to the related websites for instructions on how to install them, you can also refer to Gregorio (2010) for a skilled, though quite outdated, guide to install TEX Live on a Linux system.

[1] This concept is often mistaken with visual. It is different because "visual" means that you see fancy interfaces, use the mouse or other similar input devices; "interactive" means that the program immediately reacts to your actions and shows you updated results while you keep working. We remember of an old visual spreadsheet: Borland's Paradox. It had a switch: automatic or manual update. The second did not propagate users modifications to connected cells until an explicit update had been issued. Not that interactive.

4 LaTeX, a Macro Package Built on Top of TeX

TeX is quite a complex typesetting system and it is definitely not user friendly. Its nearly 900 commands can scare both those users who only rely on point-and-click operations and those who are not scared of using command line. Indeed those commands express the complexity of typography and the power of the TeX language (which *is* a programming language and a sort of page-description language).

Since authors are not supposed to know anything of typography—they are not typographers—it would be a bad practice asking them to use such a complex typesetting system to write their manuscripts.[2] Because of that, Leslie Lamport wrote a macro package that, while using TeX as its typesetting engine, had to provide users with a very mnemonic set of commands to structure a document. Those users—authors—had to concentrate on content, not on document look, so they just had to tag text as chapters, sections, emphasized, footnotes, and LaTeX would deploy those elements in the right way.

Going back to the *querelle* LaTeX *vs* Word, the only contact point and consequent reason to compare those tools (but it does not mean that such a comparison is meaningful) is that they both are for authors (concentrate on the content, not on the look!), though it is not immediate to find Word users who use styles. Those users who do not use styles keep on applying properties to text by hand—seldom in a coherent way. They act as typographers more than authors; should we compare Word to TeX or to InDesign and the other visual programs similar to it? We think we should not.

After studying the code of Lamport's macro package, Frank Mittelbach decided to re-program it to make it faster and less demanding when compiling a document. He also leads the LaTeX Project that provided us with LaTeX 3, a huge improvement of the language. But LaTeX is not the only macro package based on TeX. Others are ConTeXt, PDFLaTeX, XeLaTeX and LuaLaTeX (until it lasts).

While PDFLaTeX, XeLaTeX and LuaLaTeX are equivalent to LaTeX because they use at least the same set of commands, ConTeXt uses a completely different macro set. That indicates how flexible TeX is.

5 Why Text Is Better Than Binary?

Nearly every source files in the TeX ecosystem are text files (with the obvious exceptions of the compiler, external images, fonts): user documents, packages and classes, font size files, configuration files and the macro package itself (i.e., LaTeX) are text files.

A text file is a file whose content is stored as a sequence of character data. That content might be not immediately understandable (because written in a foreign language or because carries a hidden meaning) but it is surely human-readable: we recognize something resembling letters, numbers and symbols when we open it with a text editor and do not see strange, unprintable symbols or hear beeps. A binary file, on the contrary, may store its content in a more efficient way (i.e., a sequence of four-digit characters may be converted into a two-bytes integer number) but this way is usually hard to read because we do not know *a priori* the way information is stored. Humans cannot even think to read a binary file with a text editor because they could only see a sequence of strange, often unprintable, characters and symbols and hear beeps here and there (or nothing, if the picked editor does not print the BEL, ASCII symbol n. 7).

Why should a text format be picked instead of a more compact binary format? Here we list text format pros. Somebody else would instead list binary format pros and be right anyway. It depends on the task you need to accomplish.

Since a text file can be edited with whatever text editor, TeX documents can be edited even when we do not have the original editor or the compiler on our computer. It means that a TeX document can be edited with vi,[3] Emacs, Notepad, you name it, not only with a specific editor as those we will see in section 9. It can be edited on machines different than the compiling one, even running different operating systems (OS from now on), and even remote editing via telnet or ssh can be easily performed. Of course, you need TeX to generate the final document, being it PostScript or PDF, but it is unnecessary for editing the source document. Configuring the most part of LaTeX files is as easy for the same reason: the content is clear, though modifications might be a hard task for newbies.

In a professional environment it should be normal to keep track of the changes made to documents. Even better, a team might work on the same documents and it has to be possible to coordinate and integrate the job. These are the typical cases where a version control system has to be adopted. Version control systems have always correctly managed text files; they used to behave worse with binary files: being difficult to store subsequent versions as deltas, the files were entirely saved. Even though Subversion and other similar programs efficiently manage binary files (Rooney, 2005, p. 6

[2] It might even be a bad practice asking authors to write manuscripts with Adobe InDesign or Scribus. They see a white page but might find surprising not to see the cursor in the home position on that page to simply start writing. They should guess, or learn, that they can do it only if they put a text box on the page.

[3] Some readers may wonder why vi is not capitalized. Please refer to Robbins *et al.* (2008) to see that it is officially lowercased and, by the way, it is not pronounced 'six'.

and chapter 2), TeX immediately took advantage of that facility: system managers could pick the version control tool more suitable to their needs and to the needs of "their" TeX users.

A Unix filter to quickly compare a source file against another one—`diff`—can output a complete list of differences between two text files, while it will only state "differ" in case we compare two binary files. Along with this filter come some other useful tools like `sdiff`, `diff3`, `patch` that you can read about in the corresponding man pages of Unix systems or in books like Powers *et al.* (2002). A more useful tool for LaTeX users is `latexdiff` that compares two, and only two,[4] versions of a LaTeX document. It can output a document that highlights the differences between the compared files.

In conclusion we can say that text format helps TeX users to rapidly and versatilely operate on the most part of the files they work with. As we will fully understand later on, another advantage for users is that they can look at the most part of source code to get some help in writing custom commands.

6 LaTeX File Format: the Healing Text

It seems to be quite rare to find a typesetting system that stores its files as text files instead of binary files. It is understandable because every commercial vendor cares of the way it stores data and keeps them hidden from external eyes (reverse engineering is expressly forbidden...). Indeed Scribus, that stores users documents in XML, is free software. Well, as the contemporary trend indicates, XML is currently used as one of the possible file formats even for proprietary software like Microsoft Word and Adobe InDesign, so going back to a form of textually-stored data is more than a simple hope.

TeX has been on the market for about 40 years, so it has been programmed for a (computer) world that had quite narrow character sets and a wider ecosystem. Even the 8-bit ASCII set, with its 256 symbols, was too narrow to allow multilingual documents without switching between different encodings. LaTeX had quite a smart way to represent Latin extended glyphs without any switch (see section 8.1.2). The conditions are far better now because LaTeX source documents can be encoded in UTF-8, one of the Unicode text encodings and this fact widens the number of potential users and uses. Unicode formats are a kind of enlarged ASCII and the newer editors manage them without problems, provided your computer has Unicode-compliant fonts installed.

Due to the way Unicode encodes its characters, it may be possible to determine whether a file has been corrupted somehow: some mono- and multibyte sequences are not legal and cannot address to any glyph. You can read about this in Giacomelli and Pignalberi (2018) and the related bibliography. A nice and quick method to validate a UTF-8 file has been suggested in `https://stackoverflow.com/questions/115210/how-to-check-whether-a-file-is-valid-utf-8`. It is trickier to detect a corruption that transformed a legal glyph into another legal glyph. Of course, despite errors correction is not as straightforward as errors detection, some attempts to restore a corrupted text can be done: the first step is fixing the reserved part of the mono- or the multibyte character that is not legal, in the hope that the remainder part has not been corrupted itself; then a spell checker might help in case the "restored" character is in a word and this word is easy to correct; in the end, a visual inspection might be useful.

Part II: Understanding a LaTeX Document

Bonus Section: Compiling a LaTeX Document

Before discussing in detail how to write a LaTeX document, we should understand how to compile it, i.e., how to get a DVI or a PDF out of the source document. Though we will see some friendly tools in section 9 and 10, we see now the hard way.

Linux and Mac OS X users have to open a terminal; Windows users—a command interpreter. After "traveling" to the directory containing the document to be compiled (let us suppose its name is `document-name.tex`), write

```
latex document-name
```

Once the compilation successfully ends, a DVI document has been generated. This has to be post-processed with `dvips` to get a PostScript file. You might prefer to directly generate a PDF, so you are likely to use `pdflatex`, `xelatex` or `lualatex`.

For those who do not love the command line it may be easier to put the compiler icon onto the desktop and drag and drop the document icon onto the compiler icon, should the window manager support those operations.

7 The Structure of a LaTeX Document (part I)

A LaTeX document, that we know being a text file,[5] contains the complete content along with the

[4] No equivalent of `diff3` seems to exist.

[5] As we'll see in section 8.3, a LaTeX document can be subdivided in more than one text file, provided the structure coherence is preserved.

information necessary for LaTeX to typeset it. The document is composed by 1) a preliminary part of code—the preamble—containing the general typesetting rules, in the form of a class file, along with the additional commands we can use while typesetting the document and 2) by the document content—the main body or document body.

We talked about commands without introducing them. A LaTeX command is a character sequence opening with a backslash (\) followed by a word or a special symbol. It is interpreted in a special way by the typesetter. It may have 0 or more mandatory arguments along with optional arguments. When invoking the command we have to enclose the optional parameters, if any, in brackets before passing the mandatory parameters individually enclosed in braces if they are 1 or more.[6] Not all the commands are accessible to users. Part of them have been written only to be used by other commands. Commands will be a main topic along this and the subsequent lessons.

The preamble starts with the `\documentclass` command and ends when the main body starts. `\documentclass` specifies which class to use to typeset the document. The preamble is completed by using additional packages, custom commands, needed controls. In these ways we can both fine tune the general typesetting rules of the class and add unforeseen commands to accomplish tasks that are specific to our document.

The main body, enclosed in the pair of commands `\begin{document}`-`\end{document}` (TeXnically, enclosed in an *environment*), contains the document content in the form of text, command+text, environment+text, command. It sounds odd; be patient until section 8. Don't forget to notice that everything written after `\end{document}` will be neglected by LaTeX.

Since we're talking about a program that is also a programming language, we can't forget to mention that we can add comments to our source files. A comment in LaTeX starts with a percent sign (%) and lasts until the end of line.

7.1 Preamble analysis: document classes

As it is already clear, the first command in a LaTeX preamble declares which class the document is typeset according to.

The document class, which is stored in a file with extension `cls`, is a kind of configuration file containing the TeX code and the user commands necessary to typeset a specific type of documents. The class adds code, or substitutes part of it, to the macro package (i.e., LaTeX). It is mandatory to specify the class we use: without it LaTeX would not know what kind of document it has to typeset and what look it has to have.

LaTeX provides a handful of classes of general use: book, report, article, letter, slides. The Comprehensive TeX Archive Network (CTAN from now on; www.ctan.org) provides users with classes for nearly every need.

Of course, a document has to be "tailored" for the class of interest: a document initially intended to be a book can be hardly transformed in an article without restructuring it because, for instance, a book has chapters and an article has not. But it should be straightforward to compile a document initially intended to be a book into a report or a book typeset with a custom class of a specific publisher.

Let us now see a couple of examples, mutually exclusive, of `\documentclass` possible uses:

```
\documentclass{standalone}
```

```
\documentclass[a4paper,11pt]{article}
```

The first one instructs LaTeX to use a class specifically designed to arrange unstructured text or figures in the page and to crop the result. The second one tells LaTeX to typeset an article in A4-page format and with 11 points[7] text font. Examples showing documents resulting when using the above `\documentclass` instances are presented in section 8 (figures 1 and 2).

7.2 Preamble Analysis: What Is a Package?

As already mentioned, a LaTeX document could include some packages to use. The command that includes a package into a document is `\usepackage`. A package is historically a file with the `.sty` extension. Despite users can decide to store packages in files with different extension, `\usepackage` will only consider—and look for—files with `.sty` extension.

A package is a (not necessarily) small piece of code containing one or more brand new macros, one or more pre-existing but modified macros, one or more declarations. Substantially, a package is an injection of code to LaTeX so that it may rely on new features and/or on pre-existing but customized features.

Users can be asking themselves: "which is the difference between a class and a package? They seem to do the same things." It was actually true with LaTeX 2.09 (better, there were not classes at all). Since LaTeX 2_ε we may say that while a class instructs LaTeX to manage a whole document, a package is a tool to fine-tune or enhance the current behavior of the pair macro package-class.

[6] Well, it is a little more complex than this: if a parameter has more than one character, we have to group it, i.e., surround it with braces. Otherwise, only the first letter will be considered the actual parameter passed to the command.

[7] LaTeX main classes only let you pick 10, 11 or 12 points as the main text size. Other sizes are judged unsuitable for books and papers. But you are free to write a class and the corresponding `.clo` file to meet your needs.

A package is not characterized by it size: we have nearly one-liner packages (i.e., indentfirst) and 1 MB packages such as xq. Anyway, since LaTeX 2ε has been issued, the package is not a piece of software meaningful by itself: you have to use it in a more general context of a document using a class.

Also packages may have options that you declare as optional arguments to \usepackage. The manuals of the packages we intend to use will instruct us on the arguments we may set up.

7.3 Preamble Analysis: Completing the Basic Information

When dealing with LaTeX documents, we can be presented a huge series of text files. Even though currently we can be almost sure that a recent text file has been encoded in UTF-8, we cannot be as sure when managing older source files. Despite LaTeX can manage different encodings, it wants us to declare the source encoding. Not only the encoding tells how glyphs are arranged (what code number corresponds to what glyph), but also tells what OS has been used to edit the source file.[8] The package to use to specify the input encoding is inputenc. The actual encoding will be passed as the optional parameter, e.g., \usepackage[utf8]{inputenc}. This only applies to LaTeX and PDFLaTeX. XeLaTeX and LuaLaTeX assume that the input file is UTF-8 encoded.

When using LaTeX and PDFLaTeX we also have to specify the font encoding. Roughly speaking, the font encoding tells LaTeX which group of glyphs it has to use while typesetting the document. The package needed for such a specification is fontenc and the T1 encoding usually suffices. We will pass T1 as the optional parameter to \usepackage when loading fontenc. You can read the whole story in Mittelbach *et al.* (2016).

The default language used to hyphenate LaTeX documents is English. There are two packages that let us load additional or substituting languages and select them at our need: babel (normally used with LaTeX and PDFLaTeX) and polyglossia (especially programmed for XeLaTeX and LuaLaTeX). Their usage is not uniform: for instance, if you listed the languages as packages optional parameters:

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

```
\usepackage[english,italian]{polyglossia}
```

you would obtain Italian as the main language with babel and English as the main language set up by polyglossia. Unfortunately polyglossia no longer supports this technique since version 1.2. So, while the babel example is still valid, the right polyglossia example is:

```
\usepackage{polyglossia}
\setmainlanguage{italian}
\setotherlanguage{english}
```

or, in case of more than one secondary language,

```
\usepackage{polyglossia}
\setmainlanguage{italian}
\setotherlanguages{english, french}
```

More details in Charette (2015).

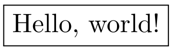
The default font used to typeset LaTeX documents is Knuth's Computer Modern (Latin Modern for XeLaTeX and LuaLaTeX). We may use another font indicating it in the preamble. While XeLaTeX and LuaLaTeX can easily use every TrueType or OpenType fonts, (PDF)LaTeX can only rely on those fonts specifically prepared for it. The best way to load the picked font is to include one of the packages especially written for that task. The presence of such a package in our systems is not a guarantee that the font file is really installed along with the package. Indeed some packages are distributed even though the font is not freely distributed. An example of such a case is URW Garamond.

Setting up the document main font to, e.g., Linux Libertine is as easy as including a package: \usepackage{libertine}. The package takes care of all the operations needed to use such a font. The manuals of such packages help us to set up the fonts according to our needs, though not every fonts have the same settings. Some of them allow us to use old style figures instead of lining figures, to allow some special ligatures and so on.

You can refer to LaTeX 3 Project Team (2005) for more information about fonts in LaTeX.

If we decide to use XeLaTeX or LuaLaTeX, we have a better and uniform control on how we set up the fonts. There is a unique package to load: fontspec, which provides us with an interface to load the fonts we prefer and decide which serif, sans serif and monospaced fonts we will use to typeset a document. The commands for doing that are \setmainfont, \setsansfont and \setmonofont; they ask for at least a mandatory argument that accepts the font name as your OS shows it (optional arguments can be used, for instance, to fine-tune OTF properties). It also allows us to link a specific font to a specific language in case we use polyglossia to set up the languages. You can refer to Robertson and Hosny (2017) to get the insights of what we only touched on and we will use in section 8.1.13.

[8] The main difference between files edited with different OSs is in the character representing the end-of-line. DOS/Windows use CR+LF (carriage return and line feed, ASCII codes 13 and 10), Mac OS until version 9 used CR, Linux and other Unix flavors use LF. In our case, we do not need to specify what OS generated the document but we have to specify what OS we are currently using to compile it. Indeed encoding files just list how and where some special characters have been encoded depending on the OS and using a different encoding results in putting strange and unwanted glyphs in our final document.



Hello, world!

FIGURE 1: The resulting PDF of the *Hello world* example.

The document preamble will also contain some data needed by some of the document kinds: title, author and date. You just need to issue the commands `\title`, `\author` and `\date` before the document main body begins. They are not automatically added to your pdf; you need to explicitly invoke the command `\maketitle` in the main body exact point you want those data to appear. Every class has a standard way to show those data. Other classes can delete or rename some of those data commands and even integrate them with other data.

European users should note that the American space after a full stop (aka period) is slightly wider than the average space between words in the same line. Since our typographic tradition is unlike the American one, we should remember to issue the `\frenchspacing` command.

8 The Structure of a LaTeX Document (part II)

8.1 Main Body Analysis: Commands for Text

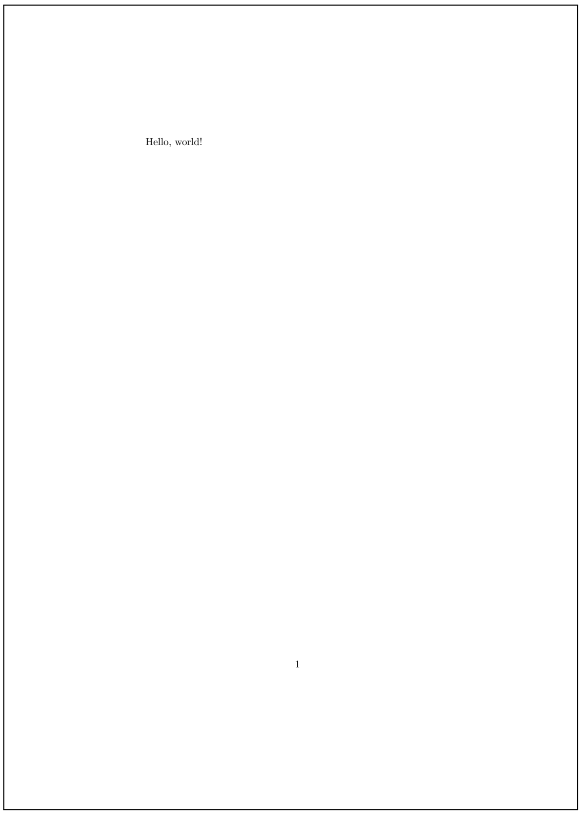
How do we write a LaTeX text? Before answering this fundamental question, we should at last see some examples that show what we discussed up to now and we will start with the usual *Hello world* example:

```
\documentclass{standalone}
\begin{document}
Hello, world!
\end{document}
```

As you may figure out, the preamble here is the only line `\documentclass{standalone}` (a command with a mandatory argument) while the main body is formed by the only line `Hello, world!`. This line only contains text and, quite comprehensibly, everything we write and do not enclose in a special command will be typeset with the main font and justified. No commands or environments have been issued so far in the document body. Let us see how the document appears, once compiled with LaTeX. That is what figure 1 shows.

Let us now change the preamble to this: `\documentclass[a4paper,11pt]{article}`. This time we passed the command a mandatory and two optional parameters. The body remains untouched. You can see the result in figure 2.

Can you see the differences? The first example generates a PDF cropped to show a tiny white frame around the text, while the second generates a PDF with the text in a specific position of an A4 page.



Hello, world!

1

FIGURE 2: The resulting PDF of the *Hello world* example with a new preamble.

The first example puts no other elements in the final PDF, while the second shows a page number. So the document class not only tells us about the look of the document related to the structure but also tells us about the document arrangement in the physical page.

8.1.1 Spaces

WPs users know, but maybe do not notice it a lot, that when they press the space bar the editor cursor shifts to right. The more they press the space bar, the farther the cursor shifts. If they ask the WP to show hidden symbols, they see a number of central dots corresponding to the times they pressed the space bar. A text with more than one space between two words shows inaccuracy.[9] That is why LaTeX disregards the spaces. Or, better, it considers more consecutive spaces as a single space.

In case a user needs to force an additional space for whatever reasons, (s)he can use the command `\␣` (i.e., a space after a backslash). The typical case is after commands like `\LaTeX` that eat spaces following the command.

Other special spaces users should know and use are the nonbreakable space (`~`), that should be used when two words are not meant to be placed in different lines, and the short (unbreakable) space

[9] Again, we are simplifying: space between words is somehow elastic because it can stretch or shrink a little bit. It is not a fixed space. Anyway, elastic or not, two spaces are wider than a single space, and that is evident.

(\,) that is shorter than the normal space and may be useful in case a text line is slightly longer than you meant or when you represent numbers with their measure unit (and are not using the siunitx package).

WPs users know that they start a new paragraph hitting return at the end of a paragraph. This is false in LaTeX, where a new paragraph starts only after a blank line. A paragraph is usually indented but in special cases or with special classes. How can we avoid to indent a paragraph? As usual in LaTeX, there are several ways and they reflect the semantic intended by the author. If the author meant not to indent a real new paragraph for whatever reason, the right way is issuing \noindent at the beginning of the paragraph the author does not want to indent. If the author just wants to interrupt the flow of a line to make the subsequent text to stand (just line an inline example) and this text is part of the paragraph, the way is issuing \\ or \newline where you he wants to terminate a line and do not leave a blank line. The new line is not in a new paragraph according to the LaTeX definition of new paragraph and just mimics it. As you can see, both ways correctly reflect the semantic of the structure.

Similarly to \newline, \newpage is useful to start a new page.

8.1.2 Special Commands for Diacritic Marks and Special Character

Even though we do not notice it when we type letters with accents and diacritic marks, possibly with a Unicode editor, TeX actually interprets commands. The original way to get an 'a with acute accent' is \'a. Can you recognize the command? Right: it is \'. The subsequent letter a is the mandatory argument. Let us now summarize (table 1) the old way to get diacritic marks and special characters with LaTeX, useful when your keyboard cannot type them directly.

Other special characters easy to get with LaTeX are dashes and quotation marks: - is a dash (-), -- is an en-dash (–), --- is an em-dash (—); `` is “, '' is ”, ` is ‘, ' is ’, << is «, >> is » (these latest quotation marks, named *guillemets* or *chevrons* in French, *virgolette caporali* in Italian, were not allowed with older font encodings like OT1). To obtain an ellipsis character is not as easy, since we cannot type three consecutive periods; we have to use the \ldots command (...).

Some other special characters that LaTeX uses, even when we do not tell him to, are the ligatures. The most common are ff, fi, fl, ffi, ffl. There are cases, like in the word *shelfful*, where the 'ff' ligature is not suitable. To avoid it we have either to write shelf\mbox{}ful or to group an f (shelf{f}ul) to get shelfful.

Two glyphs are especially able to get WPs users confused: the degree symbol and the circumflex accent (ˆ) that we have already seen. The first symbol, that older manuals advice to write as $^\circ$ (°), can be directly input with nearly any keyboard (°) or obtained via the command \textdegree provided by textcomp.[10] Users often mistake that symbol for the superscript o, which is º (textsuperscript{o})

Table 1: LaTeX commands to typeset diacritic marks and special characters.

Command	Example	Result
\\`	\\`a	à
\'	\'a	á
\^	\^o	ô
\v	\v o	ǒ
\~	\~n	ñ
\c	\c c	ç
\b	\b o	o̱
\=	\=o	ō
\u	\u a	ă
\.	\.o	ȯ
\d	\d o	ọ
\"	\"o	ö
\H	\H o	ő
\k	\k o	ǫ
\t	\t oo	o͡o
\oe		œ
\OE		Œ
\ae		æ
\AE		Æ
\aa		å
\AA		Å
\ss		ß
\o		ø
\O		Ø
\l		ł
\L		Ł
\i		ı
\j		ȷ
!\`		¡
?\`		¿

Do not forget the space in case the command is \ + letter instead of \ + mark.

With very old LaTeX versions you had to put accents on the dotless i or j, otherwise the accent would be put on the dotted characters. With more recent versions it is safe to write \'i: you will get the correct accented dotless letter.

It is possible to have the diacritic mark named comma below (r̦) and its counterpart inverted comma above (ģ) in pdfLaTeX using the *cedilla* command (\c), but only with a selected set of letters. To get the comma below under s and t you should keep using the package combelow to get such marks.

[10] The textcomp package has no manual, but you can find its symbols in Pakin (2017).

or º (`\textordmasculine`), when used to indicate an ordinal number. The second glyph—ˆ—is often mistaken for a superscript a, which is ^{a} (`\textsuperscript{a}`) or ª (`\textordfeminine`), to indicate a feminine ordinal.

Before we go on with other special characters, we invite you not to forget that LaTeX has been built on TeX and that TeX is also a programming language. Every programming language has some reserved words and some characters with special meanings. We will keep on talking of this topic in section 8.1.16.

8.1.3 Altering the Text Look

We now may want to change the look of some parts of the texts. WPs users can easily turn text (usually set up in upright serif) into italics, bold or underlined. Quite trickier for them is slanting text or turning it into small caps. So, they can (quite) easily change font shape or series. Stated that underlined is almost banned in typography, LaTeX provides us with easy commands to change the font shape or series (bold, italics, small caps, slanted). The very mnemonic commands to switch shape for a limited portion of text—that will be passed as a mandatory parameter—are:

`\textup` applies upright to the mandatory argument content;

`\textbf` applies bold to the mandatory argument content;

`\textit` applies italics to the mandatory argument content;

`\textsc` applies small caps to the mandatory argument content;

`\textsl` slants the mandatory argument content;

Please notice that these commands only work if you have all the shapes installed in your computer. Let us see a document example:

```
This is roman text.

\textbf{This is bold text.}

\textit Is this italics text?
$\leftarrow$ ???

\textit{This is italics text.}
$\leftarrow$ !!!

\textsc{This is small caps text.}

\textsl{This is slanted text.}
```

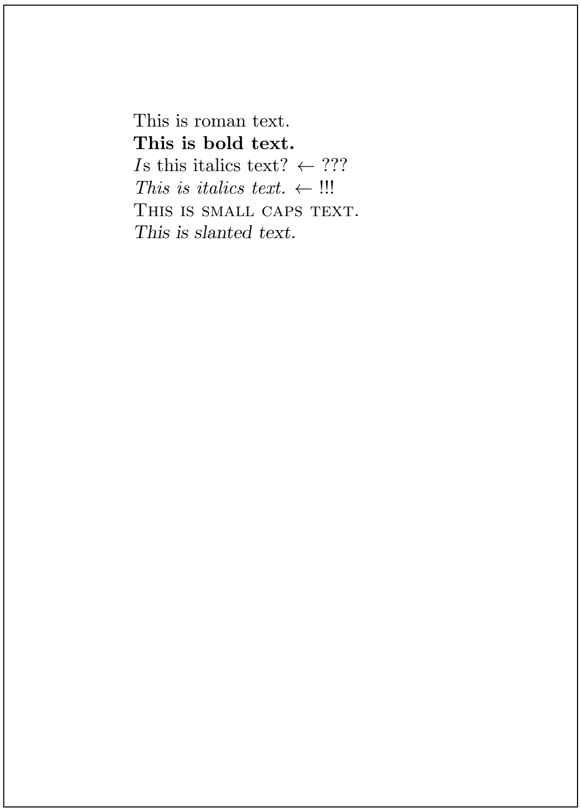

This is roman text.
This is bold text.
*I*s this italics text? ← ???
This is italics text. ← !!!
THIS IS SMALL CAPS TEXT.
This is slanted text.

FIGURE 3: Some modified text in LaTeX.

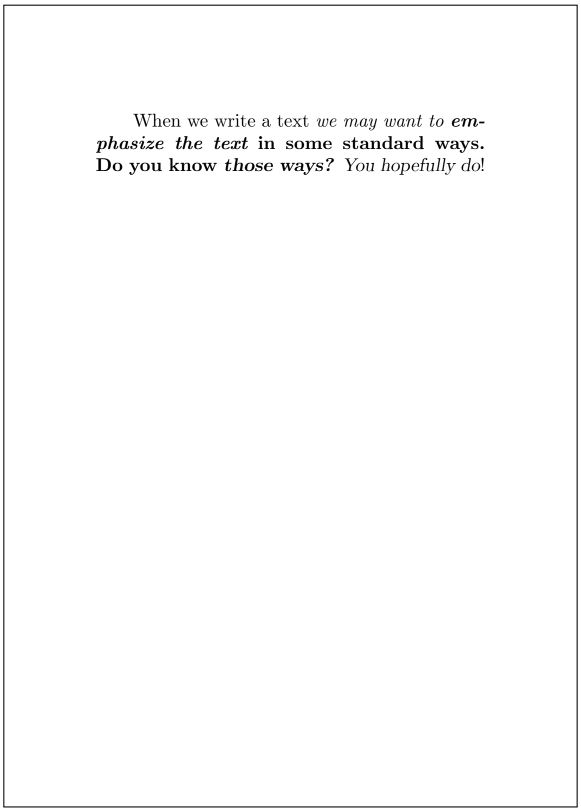
Its result is shown in figure 3. Can you explain the strange result of the third line? No? Do not worry and go reread footnote 6. If it is still unclear, the reason is that `\textit` (and similar commands) looks for a mandatory parameter. Is it our parameter longer than a character? We must surround it with braces. If we do not do that, only the first letter different from space[11] is considered the actual parameter and that is why the third line shows only the I italicized. Anyway, the important thing to understand is that the modification only applies to a defined portion of text or, to use a term common in computer science, the command has a limited scope.

Another widely used command is `\emph`: it emphasizes the mandatory argument content. The difference between it and `\textit` is that the latter always italicizes the text it is applied to while the former italicizes the upright text but uprights text in italics:
`\textit{\textit{Italics} text}` ⇒ *Italics text*
`\textit{\emph{Italics}? text}` ⇒ Italics*? text*

If we want or are forced to switch the shape or the series indefinitely, we can use these other commands: `\upshape`, `\bfseries` (`\mdseries`[12] to re-

[11] That is why we told you not to forget spaces in some cases listed in table 1.

[12] According to SCHMIDT (2006), `\mdseries` selects the regular stroke width, unlike `\bfseries` that selects the bold



When we write a text *we may want to* ***emphasize the text*** **in some standard ways. Do you know *those ways?*** *You hopefully do*!

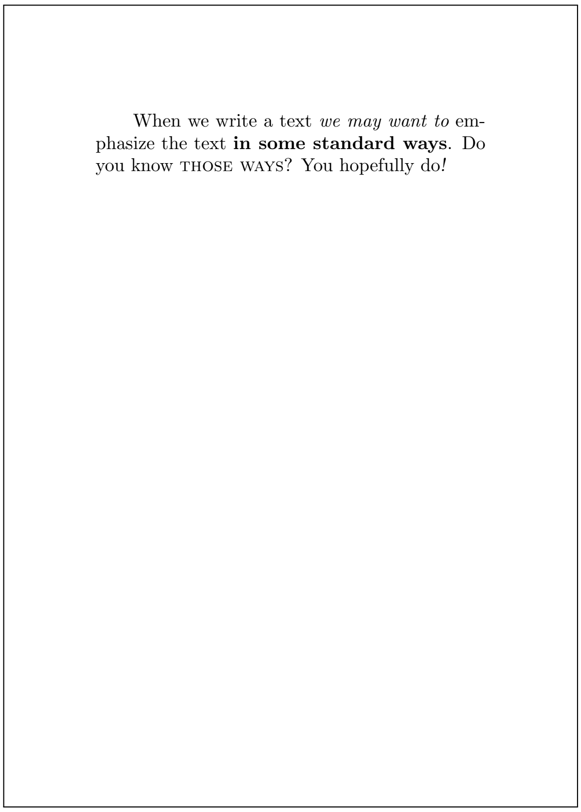
FIGURE 4: This text has been emphasized with commands without arguments.

vert it), `\itshape`, `\scshape`, `\slshape`. No argument required. The text after the issue of one such commands will be typeset according to the new shape and series until another similar command is issued. Beware the fact that a `\xxshape` adds up to a `\xxseries`, so if we issue an `\itshape` after a `\bfseries` we get the subsequent text in bold italics. On the contrary, every `\xxshape` or `\xxseries` substitutes the effects of the previously issued corresponding command. In this case, the command scope is not limited. What do you expect from the following example?

```
When we write a text
\itshape we may want to
\bfseries emphasize the text
\scshape in some standard ways.
\upshape Do you know
\slshape those ways?
\mdseries You hopefully do
\upshape!
```

The answer is in figure 4.

If you are wondering if and how is it possible to limit the scope of such commands so that you can avoid to explicitly switch back and forth, the answer is “Yes. It is possible. Just include the switching command and the text it has to be applied to into a group, i.e., include it all in braces.” The following example



When we write a text *we may want to* emphasize the text **in some standard ways**. Do you know THOSE WAYS? You hopefully do*!*

FIGURE 5: This document, apparently similar to that in figure 4, has been obtained limiting the scope of commands like `\itshape` and `\bfseries`. In this case there is no need to explicitly switch to upright roman from another shape or series.

```
When we write a text
{\itshape we may want to}
emphasize the text
{\bfseries in some standard ways}.
Do you know {\scshape those ways}?
You hopefully do{\slshape!}
```

allows the result shown in figure 5. We see that the text outside groups is upright roman because the scope of the switches we used has been limited by groups.

8.1.4 Altering the Text Font

WPs users can also easily change font family, e.g., from Times to Garamond. Since such an operation usually leads to look incoherence, LATEX lets you switch fonts in a very tricky way, but lets you easily switch from roman (serif) to sans serif or monospaced font (the TEX refers to as *typewriter typeface* or *teletype*; as we will soon see, abbreviated with tt):

> `\textrm` typesets the mandatory argument with the default serif font;
>
> `\textsf` typesets the mandatory argument with the default sans serif font;

stroke width. No way of selecting intermediate, thicker or thinner is allowed. But, of course, LATEX is not exactly a graphic design tool.

`\texttt` typesets the mandatory argument with the default monospaced font.

The corresponding commands with infinite scope are `\rmfamily`, `\sffamily` and `\ttfamily`.

Since LaTeX stresses typographical beauty, packages that load the main roman font usually also load default sans serif and teletype fonts, both picked to blend with the main font. In some cases (e.g., Concrete) the package even sets up a mathematical font (e.g., Euler). Even though this strategy seems to limit users' "creativity", it avoids typographical blasphemy, since authors should not mind how a document looks: graphic designers have already minded it. You can refer to LaTeX 3 Project Team (2005) for more information about fonts in LaTeX.

As we have seen in section 7.3, in XeLaTeX and LuaLaTeX you are free to associate whatever font to whatever family and switch to one another in the usual way, provided those fonts are installed on your computer. Beware the blend!

8.1.5 Changing the Text Shape in the Page

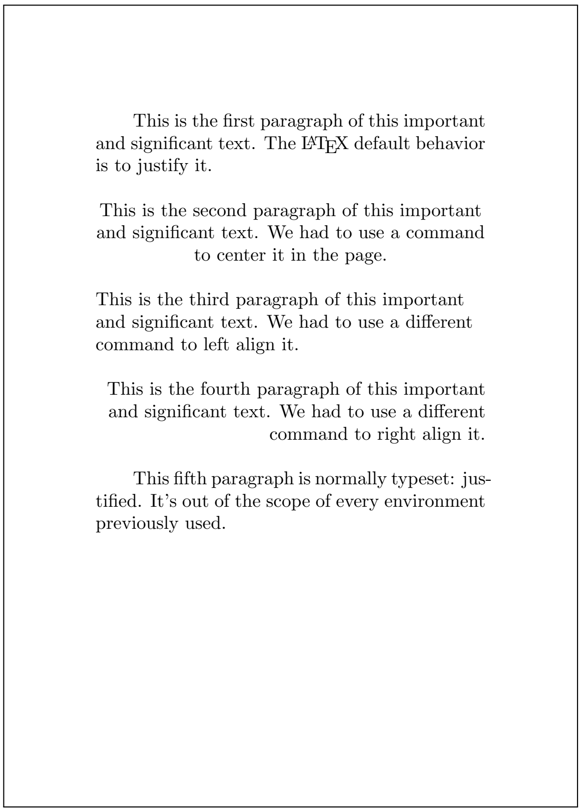
WPs users can easily change the way a text is arranged in the page. They use to start a document with the text left-aligned and have buttons to change the alignment.

Being a typesetting system mainly oriented to documents for journals, reports and books, LaTeX typesets the main text justified. Of course it is possible to change the alignment and we have two ways to accomplish that: with environments and with commands. As usual, the difference is that the environment clearly shows the scope of the change while the command affects the text from the issue point onwards.

The environments are `center`, `flushleft` (to left align) and `flushright` (to right align); the commands are `\centering`, `\raggedright` (to left align) and `\raggedleft` (to right align).

The following code, using environments, outputs the result shown in figure 6:

```
This is the first paragraph
of this important and significant
text. The \LaTeX\ default behavior
is to justify it.
\bigskip

\begin{center}
This is the second paragraph
of this important and significant
text. We had to use a command
to centering it in the page.
\end{center}
\bigskip

\begin{flushleft}
This is the third paragraph
of this important and significant
text. We had to use a different
command to left align it.
\end{flushleft}
\bigskip

\begin{flushright}
This is the fourth paragraph
of this important and significant
text. We had to use a different
command to right align it.
\end{flushright}
\bigskip

This fifth paragraph is normally
typeset: justified. It's out of
the scope of every environment
previously used.
```



This is the first paragraph of this important and significant text. The LaTeX default behavior is to justify it.

This is the second paragraph of this important
and significant text. We had to use a command
to center it in the page.

This is the third paragraph of this important and significant text. We had to use a different command to left align it.

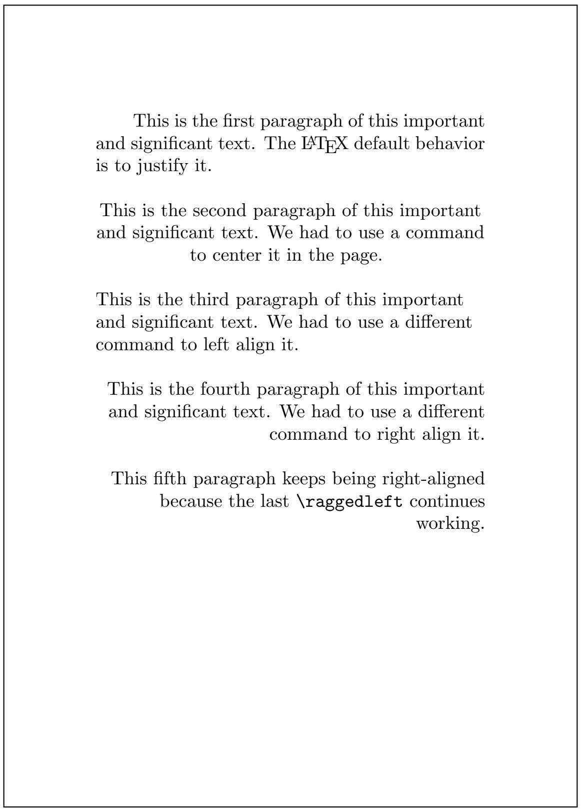
This is the fourth paragraph of this important and significant text. We had to use a different command to right align it.

This fifth paragraph is normally typeset: justified. It's out of the scope of every environment previously used.

Figure 6: Text aligning with environments.

The following code uses, instead, the commands. The result is in figure 7.

```
This is the first paragraph
of this important and significant
text. The \LaTeX\ default behavior
is to justify it.
\bigskip

\centering
This is the second paragraph
of this important and significant
text. We had to use a command
```



This is the first paragraph of this important and significant text. The LaTeX default behavior is to justify it.

This is the second paragraph of this important and significant text. We had to use a command to center it in the page.

This is the third paragraph of this important and significant text. We had to use a different command to left align it.

This is the fourth paragraph of this important and significant text. We had to use a different command to right align it.

This fifth paragraph keeps being right-aligned because the last `\raggedleft` continues working.

Figure 7: Text aligning with commands.

```
to center it in the page.
\bigskip

\raggedright
This is the third paragraph
of this important and significant
text. We had to use a different
command to left align it.
\bigskip

\raggedleft
This is the fourth paragraph
of this important and significant
text. We had to use a different
command to right align it.

\bigskip

This fifth paragraph keeps being
right-aligned because the last
\verb|\raggedleft| continues working.
```

Please, notice the blank line between the `\raggedleft`ed paragraph and the subsequent `\bigskip`. Without the blank line the command is considered part of the paragraph and the last line will end with a space, which makes it look like not being right aligned.

8.1.6 Changing the page format

Despite authors should not manage things that graphic designers should, and despite LaTeX makes not that easy to change page sizes, the geometry package (Umeki, 2010) has been programmed (especially for class programmers) to make it easier to modify whatever size you want in the page.

8.1.7 Lists

Three are the environments that LaTeX provides to write lists: itemize for bulleted lists, enumerate for numbered lists, description for labeled lists. Each list item must begin with `\item`. This command can get an optional parameter, used to replace numbers and bullets but surely fundamental to indicate labels in a labeled list.

Lists can be nested, regardless of the type but, by default, LaTeX does not allow more than three levels of depth.

8.1.8 Quoted Text, Poetry and Source Code

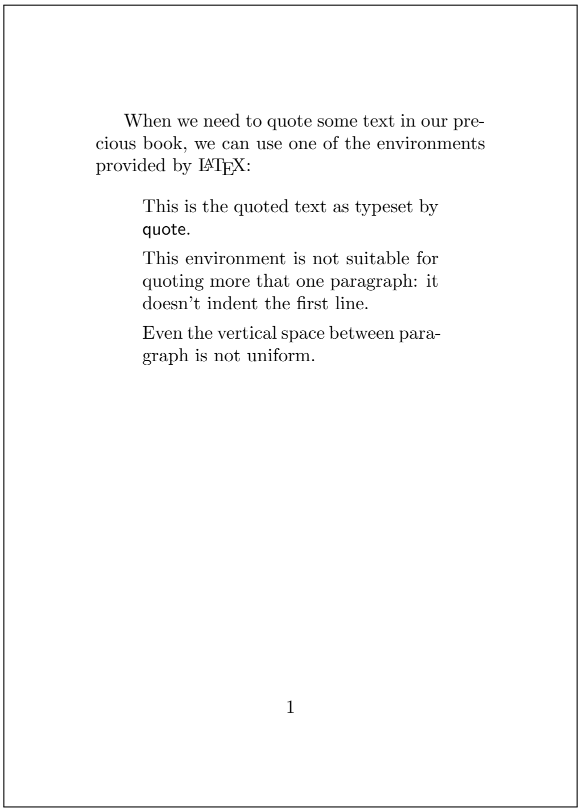
When writing books, reports, theses, we might want to quote some external contributions in the form of text excerpts. LaTeX offers two environments: quote and quotation. The first one is useful for quoting single-paragraph texts; the second one is intended for multi-paragraph texts, as it indents paragraphs, unlike quote, and the vertical space between paragraphs is equal to the vertical space between lines in a paragraph (`\baselineskip`). The text written into one of these environments (i.e., between `\begin{quote}` and `\end{quote}` or `\begin{quotation}` and `\end{quotation}`) will be quoted according to the specific style. Figures 8 and 9 show the differences.

We might even want to write poetry, where we must exactly control the interruption points. LaTeX has the suitable environment: verse.

```
When we need to write some poetry
in our precious book, we must
exactly control the interruption
points. The \textsf{verse}
environment has been written
for this case:

\begin{verse}
Stick Boy liked Match Girl,\\
He liked her a lot.\\
He liked her cute figure,\\
he thought she was hot.

But could a flame ever burn\\
for a match and a stick?\\
It did quite literally;\\
he burned up quick.
\end{verse}
```

The result is shown in figure 10.

If we need to represent some source code or a similar text, we will use the verbatim environment that typesets the text within it exactly as it is, without any interpretation.

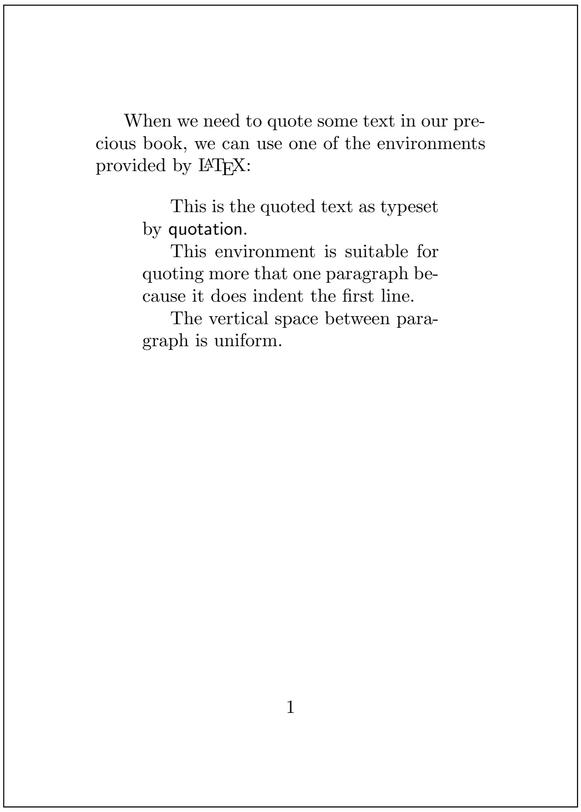


When we need to quote some text in our precious book, we can use one of the environments provided by LaTeX:

> This is the quoted text as typeset by quote.
>
> This environment is not suitable for quoting more that one paragraph: it doesn't indent the first line.
>
> Even the vertical space between paragraph is not uniform.

1

FIGURE 8: This is a multi-paragraph text quoted with quote.



When we need to quote some text in our precious book, we can use one of the environments provided by LaTeX:

> This is the quoted text as typeset by quotation.
>
> This environment is suitable for quoting more that one paragraph because it does indent the first line.
>
> The vertical space between paragraph is uniform.

1

FIGURE 9: This is a multi-paragraph text quoted with quotation.

Pay attention: you cannot nest two verbatim environments, unlike you can do with quote, quotation or verse. That is to say that you cannot show a source code containing the pair `\begin{verbatim}`-`\end{verbatim}` in a verbatim environment. That is why we chose not to include this example source code. Watch a part of it in figure 11.

If we need some verbatim terms along with the text instead of a display, we can use the command `\verb`. It works differently than usual commands: the mandatory argument will not be enclosed in braces but surrounded with a symbol (the same) that will not appear in the verbatim code (for instance, a plus sign—+—or a vertical bar—|—): `\verb+\textit{\LaTeX}+` ⇒ `\textit{\LaTeX}`

8.1.9 Footnotes

Unless you are a novel writer, with the exception of David Foster Wallace,[13] you probably need footnotes in your documents. LaTeX has a specific command, quite surprisingly named `\footnote`. You just need to issue it in the very place you want the footnote mark to appear. LaTeX takes care of correctly typesetting the footnote text (the mandatory parameter), along with the note number, at the bottom of the page.

In some special cases such as footnotes in tables, your footnotes are likely to disappear into thin air. To fix this problem you can use a pair of commands that LaTeX defines: `\footnotemark` and `\footnotetext`. The first one is intended to make the note number or symbol appear in the text; the second one is meant to make the corresponding footnote appear at the bottom of the page. Pay attention that `\footnotemark` may appear in tables, titles and other special cases but `\footnotetext` must be issued out of those structures. Moreover, you have to manually manage the footnote counter if you issues more than one `\footnotemark` before issuing the corresponding `\footnotetext`.

Despite the default footnotes look is defined in the macro package and possibly improved in classes, there are plenty of packages to modify it: footnote, footmisx, bigfoot, footmisc, just to name a few.

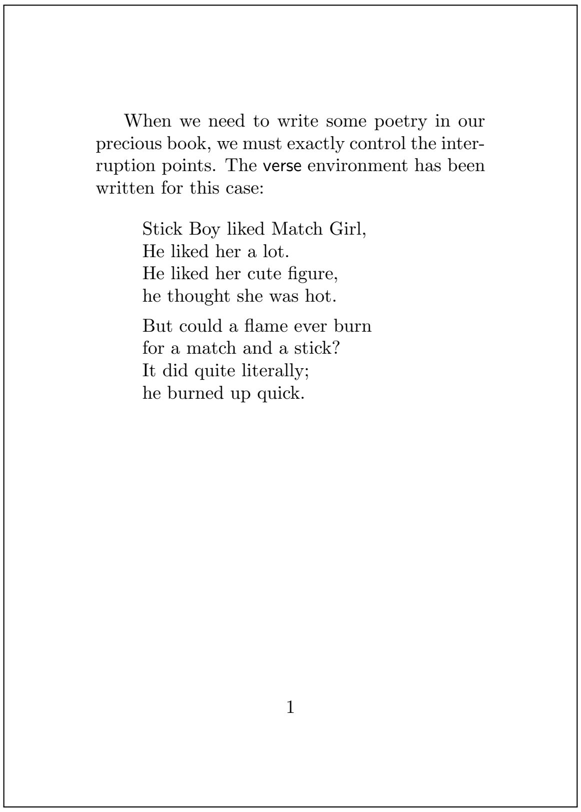
Authors may have reasons for having the footnotes grouped at the end of the documents (endnotes). While LaTeX does not provide such a mechanism, those authors have the endnotes package (LAVAGNINO, 2003) that suits their needs, though visiting CTAN may reserve other pleasing discoveries.

8.1.10 Custom Commands and Environments

In some cases we could find useful to have some custom commands to avoid extensively writing long and repetitive strings of words or to typeset specific portions of text effortlessly.

Custom environments help us define specific styles and rules for needed elements.

[13] David Foster Wallace's *Infinite Jest* is crowded of end notes: about 100 pages in 1200 pages of novel. It even has footnotes to some end notes.



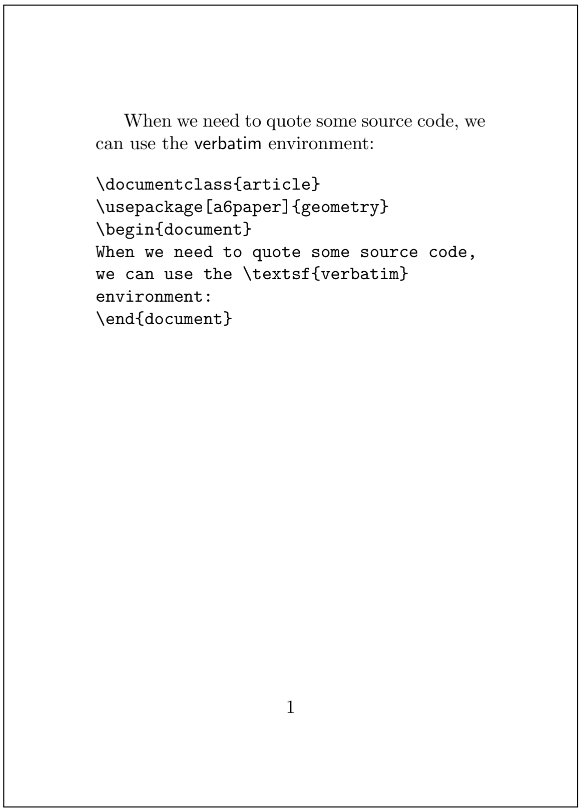
When we need to write some poetry in our precious book, we must exactly control the interruption points. The verse environment has been written for this case:

Stick Boy liked Match Girl,
He liked her a lot.
He liked her cute figure,
he thought she was hot.

But could a flame ever burn
for a match and a stick?
It did quite literally;
he burned up quick.

1

Figure 10: A document with a poem in it. The poem is Tim Burton's Stick Boy and Match Girl in Love (Burton, 1997).



When we need to quote some source code, we can use the verbatim environment:

```
\documentclass{article}
\usepackage[a6paper]{geometry}
\begin{document}
When we need to quote some source code,
we can use the \textsf{verbatim}
environment:
\end{document}
```

1

Figure 11: A document with some source code shown in it.

LaTeX provides us with the following commands: `\newcommand`, `\renewcommand`, `\newenvironment` and `\renewenvironment`, which this lesson will not discuss in details. You may find information and examples in Oetiker *et al.* (2018, pp. 104–105). Of course, the *renew* versions are meant to replace a previously defined command or environment.

Anyway, we would like you to analyze an example to understand the basic usage of one of those commands. Let us suppose we have to repetitively write the passage "All work and no play makes Jack a dull boy" in our document.[14]

```
\newcommand\Jack{All work and
no play makes Jack a dull boy}
```

The above definition of a new command tells LaTeX to typeset the text into braces whenever it sees the `\Jack` command in the document. This is the simplest use; we can include TeX code in the definition of a new command so to obtain a more complex result. You can refer, for instance, to StackExchange (2016). We can even define commands with mandatory and optional arguments. StackExchange (2011) will give you some wonderful explanations and insights about this technique.

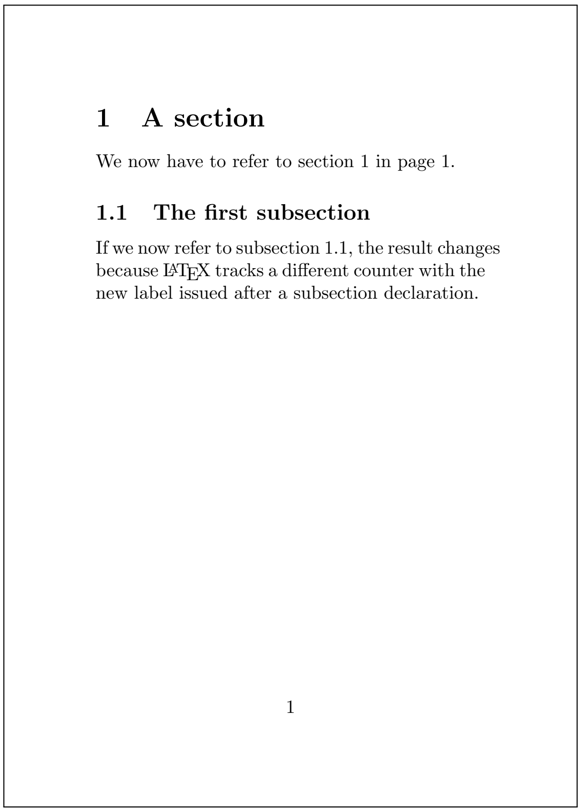
8.1.11 Tables of Contents, Cross References and Indices

We will see in section 8.2 that when we issue commands like `\chapter` and `\section` we contribute to compile a table of contents. How do we place it in our documents? It is straightforward: just write `\tableofcontents` in the place of your document where you want it to appear. Obviously, you must compile more than once to have the final pdf to be synchronized.

At last, we should talk of two other useful mechanisms that LaTeX offers the authors to write coherent yet complete documents: the cross reference and the indexing.

The cross reference is that technique that allows to refer to whatever element of a document in whatever point of the document. LaTeX has three commands to provide this technique: `\label`, `\ref` and `\pageref`. These commands have a mandatory argument that has to be a unique tag so that labels cannot be mistaken. According to their names, these commands respectively mark a point that will be referred elsewhere in the document; refer to a declared label in the form of a counter (be it a section, a figure number, a footnote. It depends on the point we issue the `\label` command); refer to a declared label in the form of a page number. The labels can be defined and referred in whatever order: they will be correctly referred only after at least the second compilation. A simple example is this:

[14] We guess that Stanley Kubrick and his crew would have loved this facility instead of typewriting that bunch of pages—even localized—for Shining!



1 A section

We now have to refer to section 1 in page 1.

1.1 The first subsection

If we now refer to subsection 1.1, the result changes because LaTeX tracks a different counter with the new label issued after a subsection declaration.

1

Figure 12: This is an example of automatic cross reference performed by LaTeX. After you correctly labeled and referenced the source document, you just have to compile the right number of times to have the right references in the final document. In this way you do not have to worry if you add some text in between: just compile again.

```
\section{A section}
\label{sec:first}
We now have to refer
to section~\ref{sec:first}
in page~\pageref{sec:first}.
\subsection{The first subsection}
\label{sub:first}
If we now refer to
subsection~\ref{sub:first},
the result changes because
\LaTeX\ tracks a different
counter with the new label
issued after a subsection
declaration.
```

and its result is in figure 12. The reference numbers are related to the points labels are issued. Our practice advises us to issue the labels to be referenced either just after the `\chapter`, `\section` and similar commands or in the mandatory argument of the `\footnote` or `\caption` commands. Those labels that will be "pagereferenced" are likely to be put in the exact points they need.

Another sensible task that an author would want to accomplish is the index compilation. Of course LaTeX has a mechanism that allows author to label every term (s)he wants in the index. After labeling, in case of changes in the text, the author just have to recompile at least a couple of times to get everything sorted out.

First of all we have to load a specific package. We strongly recommend Enrico Gregorio's imakeidx, instead of the original makeidx or the slightly more flexible splitidx. We invite you to discover the advantages of using Gregorio's package but as a teaser we can say that you no longer need to compile multiple times to have an updated index.

After we included the package, we may label every term we want with the command `\index`. This command has a mandatory argument: the term you want to appear in the index written exactly in the way you want it in the index. But you should refer to Lamport (1987) to discover the power of the LaTeX indexing mechanism and the richness of its syntax.

Now that you labeled your whole document you are ready to include the index in your document: you will issue the command `\printindex` in the point of your document you want the index to appear, compile the document and, depending on the package you are using, you might need to post-process the index file and compile your document again.

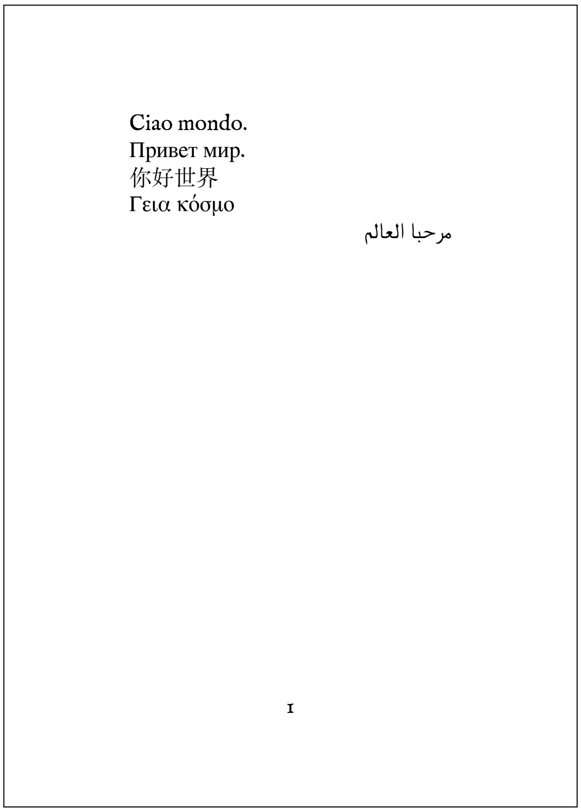
8.1.12 Arbitrary Hyphenation

It may be useful, in some special cases where words exceed the right margin, to force an hyphenation. We can do it by typing `\-` in the place we want a specific word to be hyphenated.

8.1.13 An Example of Multilingual Document in XƎLaTeX

We close this long section with an example on how easy it is for XƎLaTeX to manage languages with alphabets different from Latin. Here you can see some commands already discussed and some that you can read of in Robertson and Hosny (2017):

```
\usepackage{fontspec}
\setmainfont{IM FELL English}
\usepackage{polyglossia}
\usepackage{xeCJK}
\usepackage{bidi}
\setmainlanguage{italian}
\setotherlanguages{russian,greek,arabic}
\newfontfamily{\russianfont}{FreeSerif}
\newfontfamily{\greekfont}{FreeSerif}
\newfontfamily{\arabicfont}{FreeSerif}
\newcommand\rus[1]{%
  \foreignlanguage{russian}{#1}}
\newcommand\gre[1]{%
  \foreignlanguage{greek}{#1}}
\newcommand\ara[1]{%
  \foreignlanguage{arabic}{#1}}

\begin{document}
Ciao mondo.
```



Ciao mondo.
Привет мир.
你好世界
Γεια κόσμο
مرحبا العالم

1

FIGURE 13: The power of fontspec let us get a document like this multilingual Hello, world!.

```
\rus{Привет мир}.

你好世界

\gre{Γεια κόσμο}

\setRTL\ara{ ملاعلا ابحرم}
```

Figure 13 shows the related result.

We specified XƎLaTeX because LuaLaTeX has different mechanisms and packages to manage Right-to-Left languages. For instance, LuaLaTeX is not compatible with bidi, so the previous example has to be modified to be compiled with LuaLaTeX.

8.1.14 Floating bodies: figures and tables

Depending on the document we are writing, we might have to integrate the text with images and tables. Despite way too many authors want these elements put exactly where they think they should be, LaTeX encourages us to let them float, so that it can keep typesetting the pages according to its quality high standard.

How do we make them float? It suffices to put them in a floating body. LaTeX provides us with two environments that are floating bodies: figure and table. In such environments we will respectively put graphics, drawings (both discussed in a different lesson), images (with the command `\includegraphics` provided by graphicx; this is an extension of the former graphics package) and tables. As for images, graphicx enables us to include JPG, PNG, PDF and EPS images and allow to rotate, clip, trim, dimension and scale images. You may find a complete manual in DAVID P. CARLISLE AND THE LATEX3 PROJECT (2017).

The very basic yet complete usage is

```
\begin{figure}
\centering
\includegraphics{image.pdf}
\caption{Image caption.}
\end{figure}
```

which shows how we add a horizontally centered floating image and caption to our documents putting those elements into a figure container.

Like the previous case, table is just a container that can contain anything, most likely tables. To build a table we need a constructor and the LaTeX basic table constructor is the tabular environment. Instead of explaining, let us see an example, though shortened, related to table 1 and shown in table 2:

```
\begin{table}
\caption{\label{tab:dm-
ex}An example of table construction.}
\begin{tabular}{|c|c|c|}
\hline
\textsc{Command} & \textsc{Example} &
  \textsc{Result} \\
\hline
\verb|\`| & \verb|\`a| & \`a \\
\verb|\'| & \verb|\'a| & \'a \\
\hline
\multicolumn{3}{p{.93\columnwidth}}{It's
possible to have the diacritic mark named
comma below (\c{r}) and its counterpart
inverted comma above (\c{g}) in
\pdfLaTeX\ using the \emph{cedilla}
command (\cmdname{c}), but only with a
selected set of letters. To get the
comma below under s and t you should
keep using the package \pkgname{combelow}
to get such marks.} \\
\hline
\end{tabular}
\end{table}
```

We already know that the pair `\begin{table}-\end{table}` is the floating body that lets the table be placed where LaTeX considers the better place; we also know that the pair `\begin{tabular}-\end{tabular}` is the construction environment. `\begin{tabular}` needs a mandatory argument that specifies how many columns compose the table and their alignment. In our case we see three `c` interleaved by vertical bars; that means 'three centered columns delimited by vertical rules'. Other alignments are `l` (for left), `r` (for right) and `p{`⟨*dimension*⟩`}` (for left aligned in a fixed width column). We can leave out vertical

Table 2: An example of table construction.

Command	Example	Result
`\``	`\`a`	à
`\'`	`\'a`	á

It is possible to have the diacritic mark named comma below (ŗ) and its counterpart inverted comma above (ģ) in pdfLaTeX using the *cedilla* command (`\c`), but only with a selected set of letters. To get the comma below under s and t you should keep using the package combelow to get such marks.

Table 3: LaTeX commands to give documents a structure.

Command	book	report	article
`\part`	✓	✓	✓
`\chapter`	✓	✓	
`\section`	✓	✓	✓
`\subsection`	✓	✓	✓
`\subsubsection`	✓	✓	✓
`\paragraph`	✓	✓	✓
`\subparagraph`	✓	✓	✓

rules, if we do not want them. The command `\hline` draws horizontal rules.

It is now time to fill the cells. We start writing the content of the first cell (upper left). The content ends when we issue a &. After filling the last cell in a table row we will not issue a & but a line break (\\).

Now, before you ask us why tables 1 and 2 look so different, we should unveil that table 1 has been drawn using tabu, a package that allows a finer control over the table design and look, and booktabs, a package that provides us with finer rulers and better vertical spacing.

The caption has to be issued with the command `\caption` before we open tabular or after we close it. Please notice the `\label` in its parameter so that the related reference-counter is correctly referred to a table number.

8.1.15 Colors

LaTeX is not bound to serious black text and white background. Thanks to packages like color or xcolor we can add colors to text, highlight it coloring its background and color a whole page so that the text is on a background different than white. While the color manual is in the graphicx manual (David P. Carlisle and The LaTeX3 Project, 2017), xcolor has a different manual that you can read, as usual, issuing the command `texdoc xcolor` in a terminal.

8.1.16 Again on Special Characters

Now we have a clearer idea of how LaTeX uses some characters: \ starts a command, so it cannot be used *per se* to represent a backslash; `~` is a special space, so it cannot be used to represent a tilde character; `&` is a tabbing character in tables and cannot represent the ampersand character; `%` starts a comment.

Tilde can be obtained as `\~` (˜, maybe not exactly in the position we expected it), as `$\sim$` (∼, quite big, huh?) and as `\textasciitilde` via textcomp (˜, like in the first case). Ampersand is `\&` and percent is `\%`.

What about the backslash, since `\\` breaks a line? Once again textcomp helps: `\textbackslash` (\).

8.2 Main Body Analysis: Document Structure

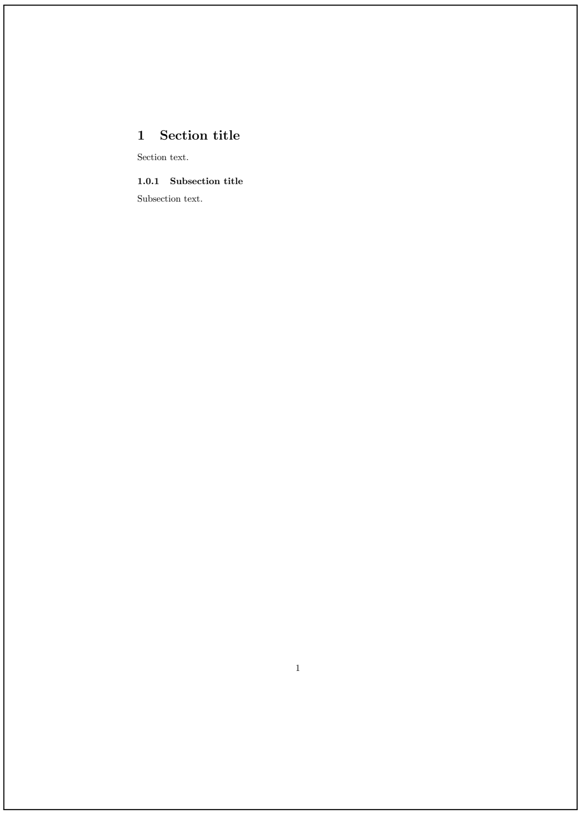
The structure of a document is directly related to the type of the document. As we have mentioned in section 7.1, the way a book is organized is slightly different than the way an article is organized, and both of them totally differ from a letter. Roughly speaking, a document structure is directly related to the text organization.

When we organize a document, being it a book, a report or an article, we subdivide it into several main topics (chapters for books and reports, sections for articles). They could be grouped into parts, but not necessarily. This leads to a special treatment of counters that we are going to talk about in a few lines. A main part will probably be subdivided into smaller blocks of text, each one treating a specific subtopic: sections for books and reports, subsections for articles. Every smaller block can be subdivided into smaller subblocks. When we organize our documents according to these blocks and subblocks—levels—, we are giving our documents a structure. Table 3 lists the commands provided by LaTeX to structure documents and indicates which classes use them or not.

All of these commands take a mandatory argument (the title) and an optional argument (the title as it will be included in the table of contents. If no optional argument has been indicated, the main title will be included in the table of contents). The commands add a number before every title. Unless we change the way the numbers are stored, they have the following forms: part number, chapter number, [chapter.]section number (the brackets mean that there is no chapter numbers in an article), [chapter.]section.subsection number and so on.

As you can figure out, this "strategy" helps authors not to make errors with the structure. Suppose we are writing an article. After the `\section`, instead of declaring a `\subsection`, we declare a `\subsubsection` as in the following example:

```
\begin{document}
\section{Section title}
```



1 Section title

Section text.

1.0.1 Subsection title

Subsection text.

1

Figure 14: The number 1.0.1 indicates that we skipped a level in the structure.

```
Section text.

\subsubsection{Subsection title}
Subsection text.
\end{document}
```

As we can see in figure 14, what we expected to be numbered 1.1 is instead numbered 1.0.1. This means that we skipped a level: instead of using a `\subsection` command, we used a `\subsubsection`.

All of the commands we mentioned in this section have a corresponding starred version (i.e., `\section*`). These versions do not print the number before the corresponding title and do not let the titles appear in the table of contents.

It is a good practice, when we organize the structure, to let (let us say) a section have two or more subsections, if subsections are necessary.

8.3 Splitting Big Documents

When writing big documents such as books, we could find handy to split the LaTeX source into different files. How do we put them all together?

The most common case is a master file that includes slave documents, for instance one file per chapter in the document body.

LaTeX offers two ways to include such files: `\include{`⟨*filename*⟩`}` (without the `.tex` extension) starts a new page, separately processes the file and includes it in the final camera ready; `\input{`⟨*filename*⟩`}` (without the `.tex` extension) that simply processes the file as being part of a single, big file.

You will find further information on the fine control of such an operation in Oetiker *et al.* (2018) or, better, in Kopka and Daly (2004).

8.4 Help, I Need a Symbol

If you spent many hours writing LaTeX documents, you may have experienced that you cannot recall the name of a specific symbol you really need now. Reading Pakin (2017) could be helpful, but you are not sure you can really locate that symbol in a manual longer than 300 pages. How can you do?

Thanks to Philipp Kühl and Daniel Kirsch, we may rely on a website (Kühl and Kirsch, 2019) that allows us to sketch the symbol we are looking for and answers with a list of those symbols that look like our sketch along with the related commands.

Bonus Section: Guess What!

Many of you, experienced with LaTeX, know the usual aspect of LaTeX documents. You even know the power and the lacks of TeX. So we decided to deliver you a game: figures 15–31 show some books, journals, magazines and report pages along with other documents. You are invited to guess which of them have been typeset with LaTeX and which of them have not been. The solution is at the end of the lesson.

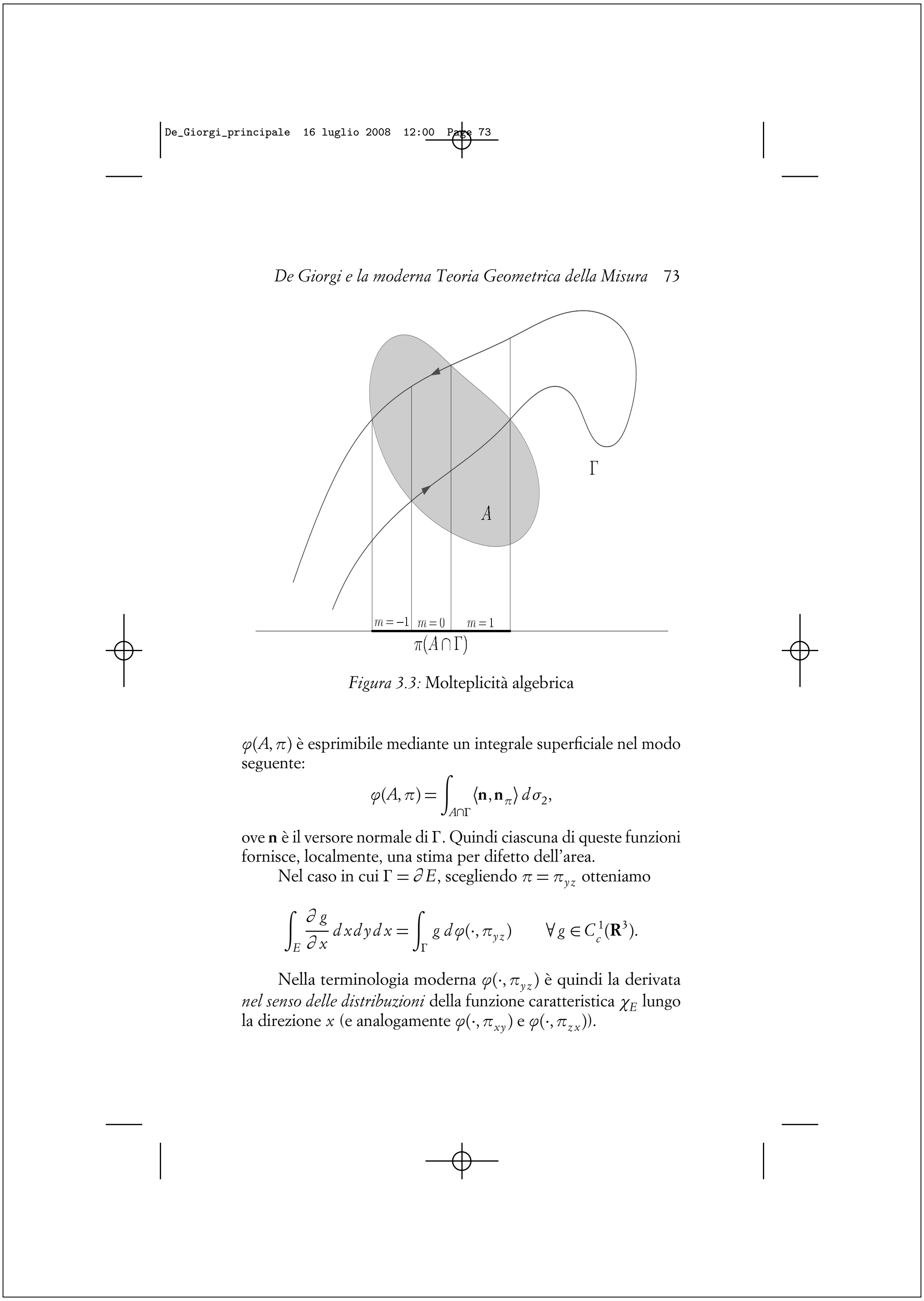
Part III: Writing LaTeX Documents

9 (Not Necessarily) Dedicated Editors

We already know that users do not need a special editor to write a LaTeX document. We will not stress enough the fact that users tend to think in terms of tools; the risk is that a user who normally edits LaTeX documents with, say, Kile, gets lost without it and does not consider to use a different IDE or text editor or even does not consider the fact that a different editor can be used (can we call it the Word syndrome?).[15] In spite of that, LaTeX users can rely on several more or less dedicated editors.

The first one, one of the most famous tools in the Unix world, is Emacs. It is not especially written to integrate with LaTeX, but with AUCTeX it can be used as a powerful IDE (integrated development

[15] [NdGP] When I used to write programs with Borland Turbo Pascal and Turbo C, I could not understand that I was allowed to edit source codes with whatever editor, so I always opened the IDE not only to get an executable program but even because I thought I could not edit the source code in a different way, just like any proprietary environment. Then I switched to Unix...



De_Giorgi_principale 16 luglio 2008 12:00 Page 73

De Giorgi e la moderna Teoria Geometrica della Misura 73

Figura 3.3: Molteplicità algebrica

$\varphi(A,\pi)$ è esprimibile mediante un integrale superficiale nel modo seguente:

$$\varphi(A,\pi)=\int_{A\cap\Gamma}\langle \mathbf{n},\mathbf{n}_{\pi}\rangle\, d\sigma_2,$$

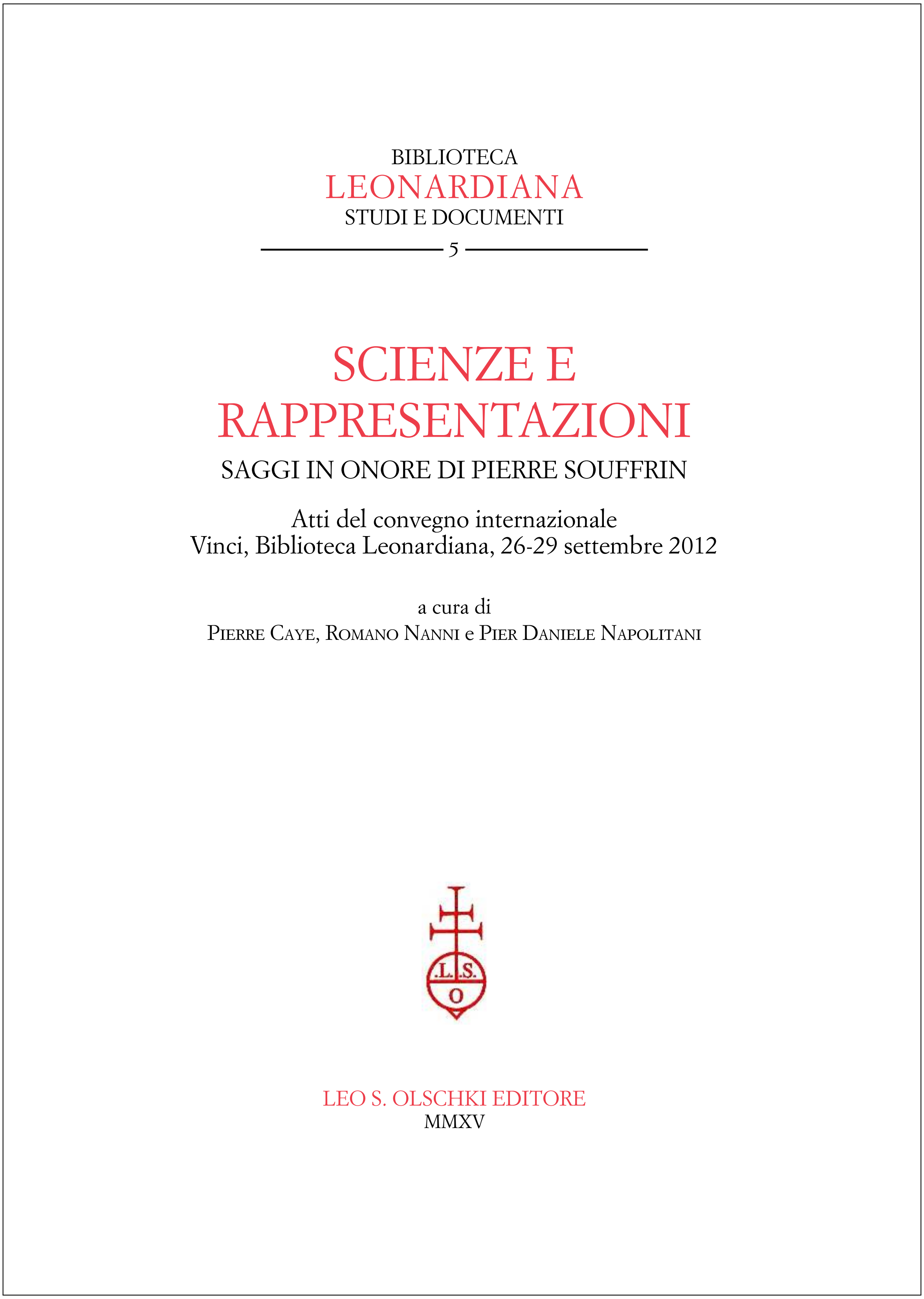
ove **n** è il versore normale di Γ. Quindi ciascuna di queste funzioni fornisce, localmente, una stima per difetto dell'area.

Nel caso in cui $\Gamma=\partial E$, scegliendo $\pi=\pi_{yz}$ otteniamo

$$\int_E \frac{\partial g}{\partial x}\, dxdydx=\int_\Gamma g\, d\varphi(\cdot,\pi_{yz}) \qquad \forall g\in C_c^1(\mathbf{R}^3).$$

Nella terminologia moderna $\varphi(\cdot,\pi_{yz})$ è quindi la derivata *nel senso delle distribuzioni* della funzione caratteristica χ_E lungo la direzione x (e analogamente $\varphi(\cdot,\pi_{xy})$ e $\varphi(\cdot,\pi_{zx})$).

FIGURE 15: Mathematical formulae and diagrams.



BIBLIOTECA
LEONARDIANA
STUDI E DOCUMENTI

5

SCIENZE E
RAPPRESENTAZIONI

SAGGI IN ONORE DI PIERRE SOUFFRIN

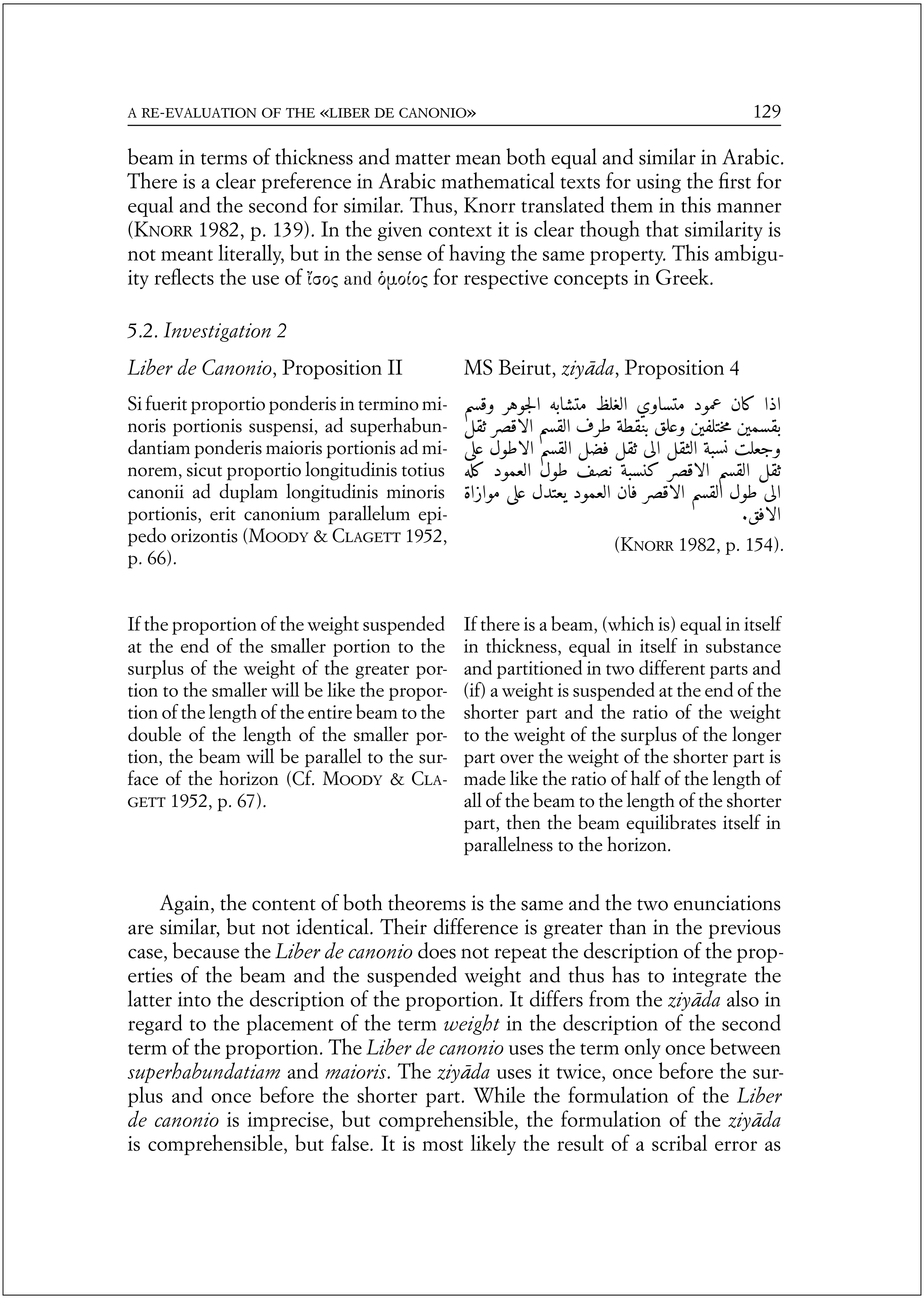
Atti del convegno internazionale
Vinci, Biblioteca Leonardiana, 26-29 settembre 2012

a cura di
Pierre Caye, Romano Nanni e Pier Daniele Napolitani

L.S.
O

LEO S. OLSCHKI EDITORE
MMXV

Figure 16: Frontispiece of a proceedings volume, published by Olschki.



A RE-EVALUATION OF THE «LIBER DE CANONIO» 129

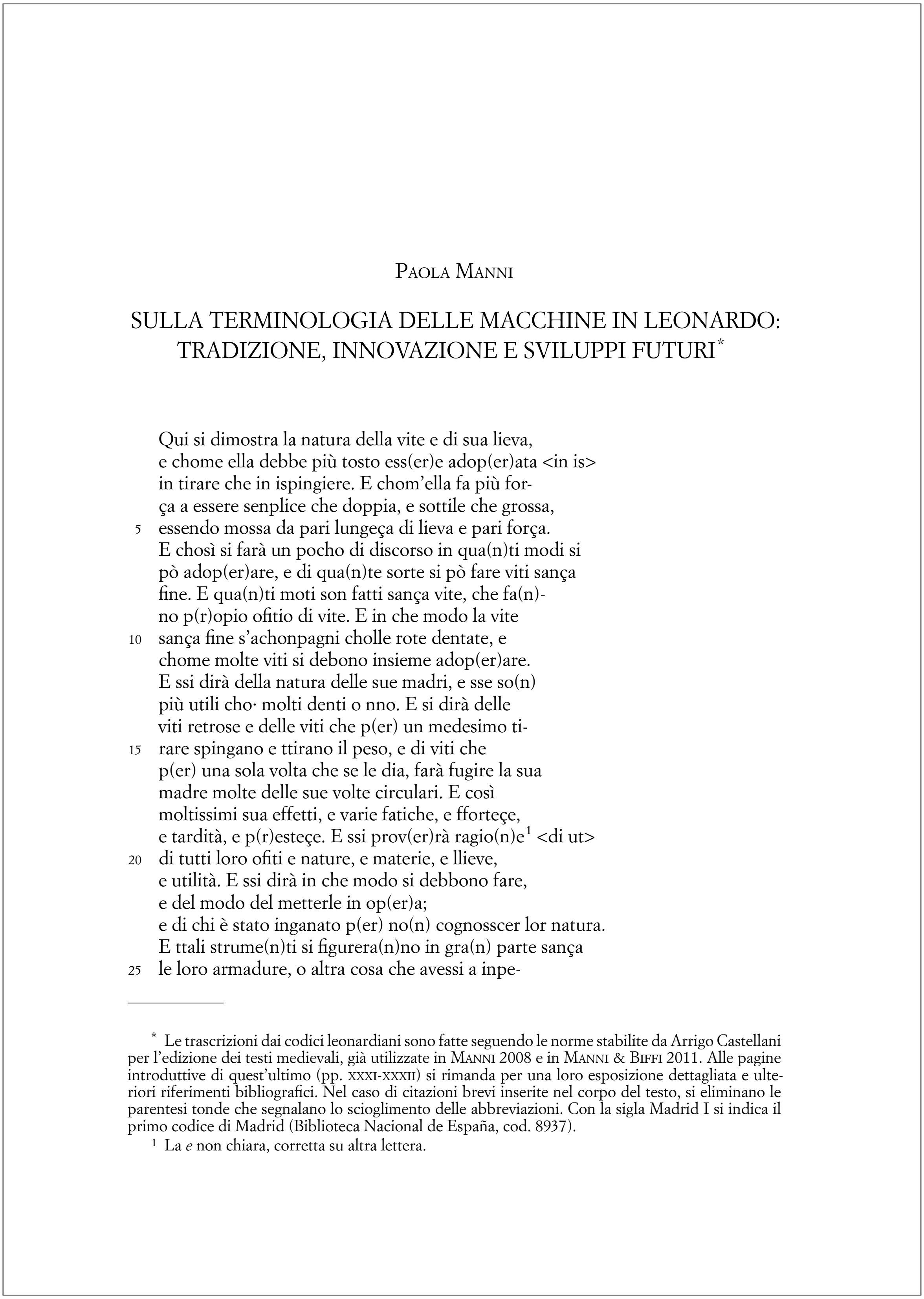
beam in terms of thickness and matter mean both equal and similar in Arabic. There is a clear preference in Arabic mathematical texts for using the first for equal and the second for similar. Thus, Knorr translated them in this manner (KNORR 1982, p. 139). In the given context it is clear though that similarity is not meant literally, but in the sense of having the same property. This ambiguity reflects the use of ἴσος and ὁμοίος for respective concepts in Greek.

5.2. *Investigation 2*

Liber de Canonio, Proposition II	MS Beirut, *ziyāda*, Proposition 4
Si fuerit proportio ponderis in termino minoris portionis suspensi, ad superhabundantiam ponderis maioris portionis ad minorem, sicut proportio longitudinis totius canonii ad duplam longitudinis minoris portionis, erit canonium parallelum epipedo orizontis (MOODY & CLAGETT 1952, p. 66).	اذا كان عمود متساوي الغلظ متشابه الجوهر وقسم بقسمين مختلفين وعلق بنقطة طرف القسم الاقصر ثقل وجعلت نسبة الثقل الى ثقل فضل القسم الاطول على ثقل القسم الاقصر كنسبة نصف طول العمود كله الى طول القسم الاقصر فان العمود يعتدل على موازاة الافق. (KNORR 1982, p. 154).
If the proportion of the weight suspended at the end of the smaller portion to the surplus of the weight of the greater portion to the smaller will be like the proportion of the length of the entire beam to the double of the length of the smaller portion, the beam will be parallel to the surface of the horizon (Cf. MOODY & CLAGETT 1952, p. 67).	If there is a beam, (which is) equal in itself in thickness, equal in itself in substance and partitioned in two different parts and (if) a weight is suspended at the end of the shorter part and the ratio of the weight to the weight of the surplus of the longer part over the weight of the shorter part is made like the ratio of half of the length of all of the beam to the length of the shorter part, then the beam equilibrates itself in parallelness to the horizon.

Again, the content of both theorems is the same and the two enunciations are similar, but not identical. Their difference is greater than in the previous case, because the *Liber de canonio* does not repeat the description of the properties of the beam and the suspended weight and thus has to integrate the latter into the description of the proportion. It differs from the *ziyāda* also in regard to the placement of the term *weight* in the description of the second term of the proportion. The *Liber de canonio* uses the term only once between *superhabundatiam* and *maioris*. The *ziyāda* uses it twice, once before the surplus and once before the shorter part. While the formulation of the *Liber de canonio* is imprecise, but comprehensible, the formulation of the *ziyāda* is comprehensible, but false. It is most likely the result of a scribal error as

FIGURE 17: Multilingual parallel texts, from the same volume.



Paola Manni

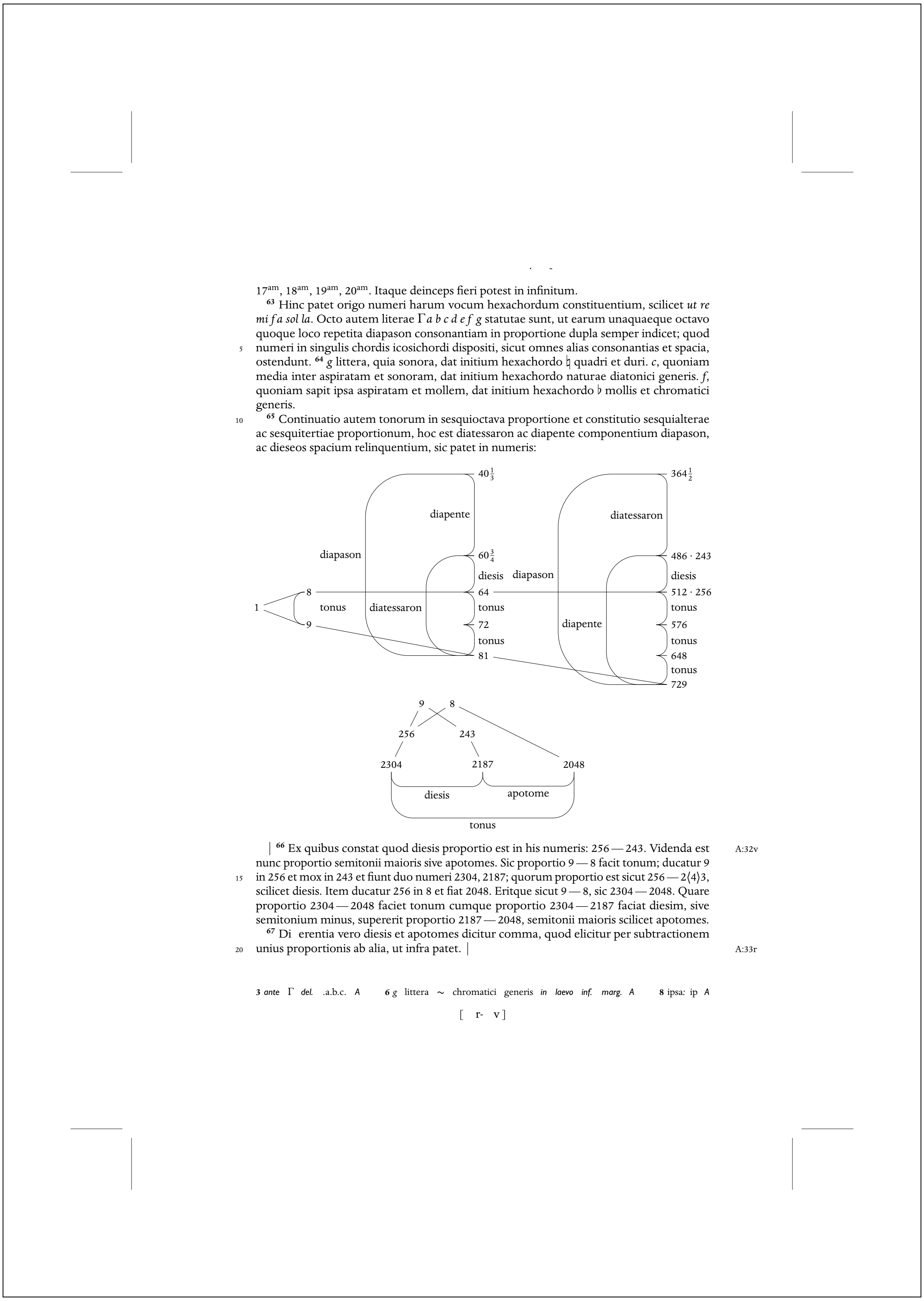
SULLA TERMINOLOGIA DELLE MACCHINE IN LEONARDO: TRADIZIONE, INNOVAZIONE E SVILUPPI FUTURI[*]

Qui si dimostra la natura della vite e di sua lieva,
e chome ella debbe più tosto ess(er)e adop(er)ata <in is>
in tirare che in ispingiere. E chom'ella fa più for-
ça a essere senplice che doppia, e sottile che grossa,
5 essendo mossa da pari lungeça di lieva e pari força.
E chosì si farà un pocho di discorso in qua(n)ti modi si
pò adop(er)are, e di qua(n)te sorte si pò fare viti sança
fine. E qua(n)ti moti son fatti sança vite, che fa(n)-
no p(r)opio ofitio di vite. E in che modo la vite
10 sança fine s'achonpagni cholle rote dentate, e
chome molte viti si debono insieme adop(er)are.
E ssi dirà della natura delle sue madri, e sse so(n)
più utili cho· molti denti o nno. E si dirà delle
viti retrose e delle viti che p(er) un medesimo ti-
15 rare spingano e ttirano il peso, e di viti che
p(er) una sola volta che se le dia, farà fugire la sua
madre molte delle sue volte circulari. E così
moltissimi sua effetti, e varie fatiche, e fforteçe,
e tardità, e p(r)esteçe. E ssi prov(er)rà ragio(n)e[1] <di ut>
20 di tutti loro ofiti e nature, e materie, e llieve,
e utilità. E ssi dirà in che modo si debbono fare,
e del modo del metterle in op(er)a;
e di chi è stato inganato p(er) no(n) cognosscer lor natura.
E ttali strume(n)ti si figurera(n)no in gra(n) parte sança
25 le loro armadure, o altra cosa che avessi a inpe-

[*] Le trascrizioni dai codici leonardiani sono fatte seguendo le norme stabilite da Arrigo Castellani per l'edizione dei testi medievali, già utilizzate in Manni 2008 e in Manni & Biffi 2011. Alle pagine introduttive di quest'ultimo (pp. xxxi-xxxii) si rimanda per una loro esposizione dettagliata e ulteriori riferimenti bibliografici. Nel caso di citazioni brevi inserite nel corpo del testo, si eliminano le parentesi tonde che segnalano lo scioglimento delle abbreviazioni. Con la sigla Madrid I si indica il primo codice di Madrid (Biblioteca Nacional de España, cod. 8937).

[1] La *e* non chiara, corretta su altra lettera.

Figure 18: Automatic line numbering, from the same volume.



17^{am}, 18^{am}, 19^{am}, 20^{am}. Itaque deinceps fieri potest in infinitum.

63 Hinc patet origo numeri harum vocum hexachordum constituentium, scilicet *ut re mi fa sol la*. Octo autem literae Γ *a b c d e f g* statutae sunt, ut earum unaquaeque octavo quoque loco repetita diapason consonantiam in proportione dupla semper indicet; quod numeri in singulis chordis icosichordi dispositi, sicut omnes alias consonantias et spacia, ostendunt. **64** *g* littera, quia sonora, dat initium hexachordo ♮ quadri et duri. *c*, quoniam media inter aspiratam et sonoram, dat initium hexachordo naturae diatonici generis. *f*, quoniam sapit ipsa aspiratam et mollem, dat initium hexachordo ♭ mollis et chromatici generis.

65 Continuatio autem tonorum in sesquioctava proportione et constitutio sesquialterae ac sesquitertiae proportionum, hoc est diatessaron ac diapente componentium diapason, ac dieseos spacium relinquentium, sic patet in numeris:

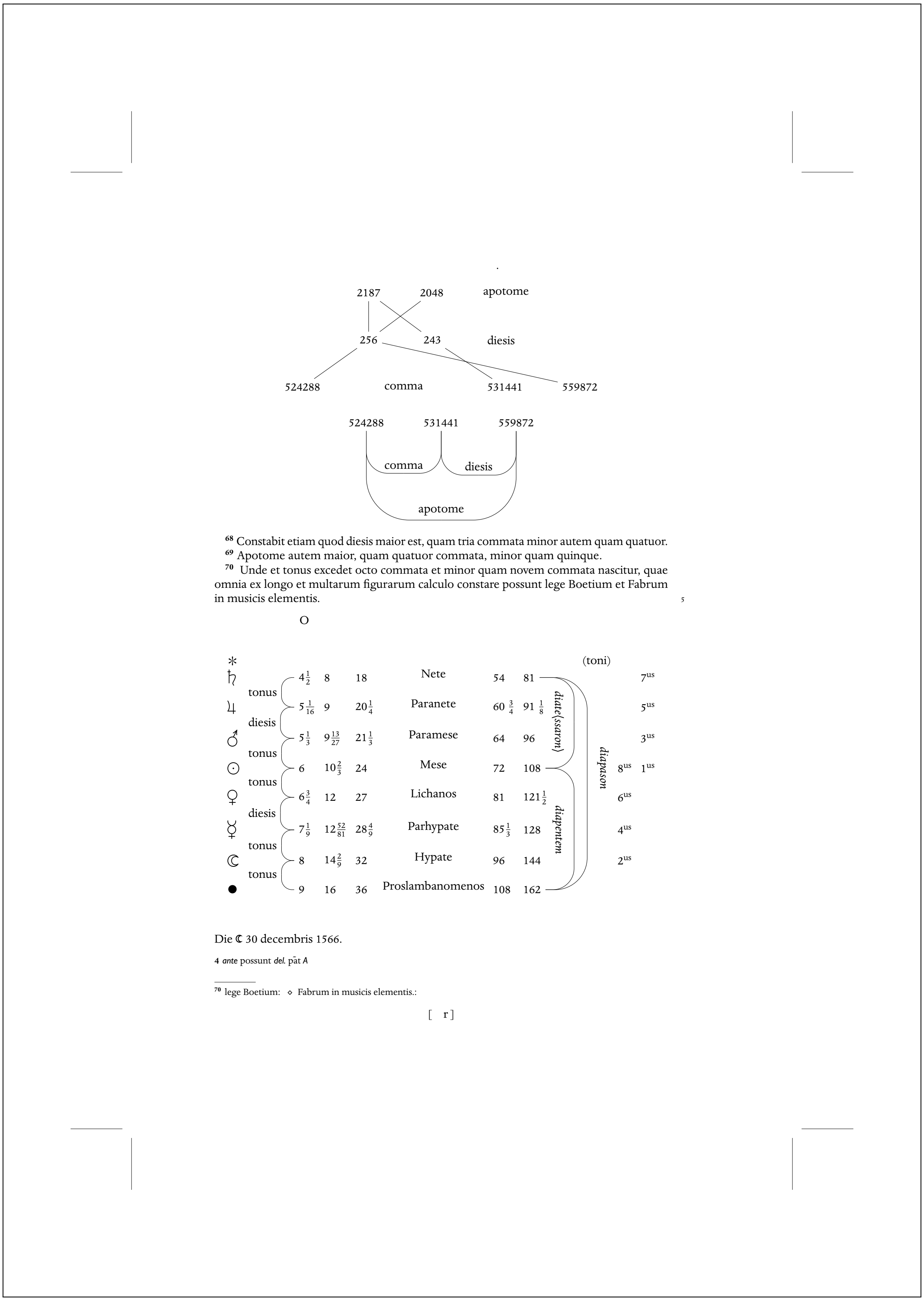
| **66** Ex quibus constat quod diesis proportio est in his numeris: 256 — 243. Videnda est nunc proportio semitonii maioris sive apotomes. Sic proportio 9 — 8 facit tonum; ducatur 9 in 256 et mox in 243 et fiunt duo numeri 2304, 2187; quorum proportio est sicut 256 — 2⟨4⟩3, scilicet diesis. Item ducatur 256 in 8 et fiat 2048. Eritque sicut 9 — 8, sic 2304 — 2048. Quare proportio 2304 — 2048 faciet tonum cumque proportio 2304 — 2187 faciat diesim, sive semitonium minus, supererit proportio 2187 — 2048, semitonii maioris scilicet apotomes. A:32v

67 Differentia vero diesis et apotomes dicitur comma, quod elicitur per subtractionem unius proportionis ab alia, ut infra patet. | A:33r

3 *ante* Γ *del.* .a.b.c. *A* **6** *g* littera ∼ chromatici generis *in laevo inf. marg. A* **8** ipsa*:* ip *A*

[r- v]

Figure 19: Diagrams from the critical edition of Francesco Maurolico's *Musica*.



[68] Constabit etiam quod diesis maior est, quam tria commata minor autem quam quatuor.
[69] Apotome autem maior, quam quatuor commata, minor quam quinque.
[70] Unde et tonus excedet octo commata et minor quam novem commata nascitur, quae omnia ex longo et multarum figurarum calculo constare possunt lege Boetium et Fabrum in musicis elementis. 5

O

*								(toni)	
♄		$4\frac{1}{2}$	8	18	Nete	54	81		7^{us}
	tonus								
♃		$5\frac{1}{16}$	9	$20\frac{1}{4}$	Paranete	$60\ \frac{3}{4}$	$91\ \frac{1}{8}$		5^{us}
	diesis								
♂		$5\frac{1}{3}$	$9\frac{13}{27}$	$21\frac{1}{3}$	Paramese	64	96		3^{us}
	tonus								
⊙		6	$10\frac{2}{3}$	24	Mese	72	108	8^{us}	1^{us}
	tonus								
♀		$6\frac{3}{4}$	12	27	Lichanos	81	$121\frac{1}{2}$	6^{us}	
	diesis								
☿		$7\frac{1}{9}$	$12\frac{52}{81}$	$28\frac{4}{9}$	Parhypate	$85\frac{1}{3}$	128	4^{us}	
	tonus								
☾		8	$14\frac{2}{9}$	32	Hypate	96	144	2^{us}	
	tonus								
●		9	16	36	Proslambanomenos	108	162		

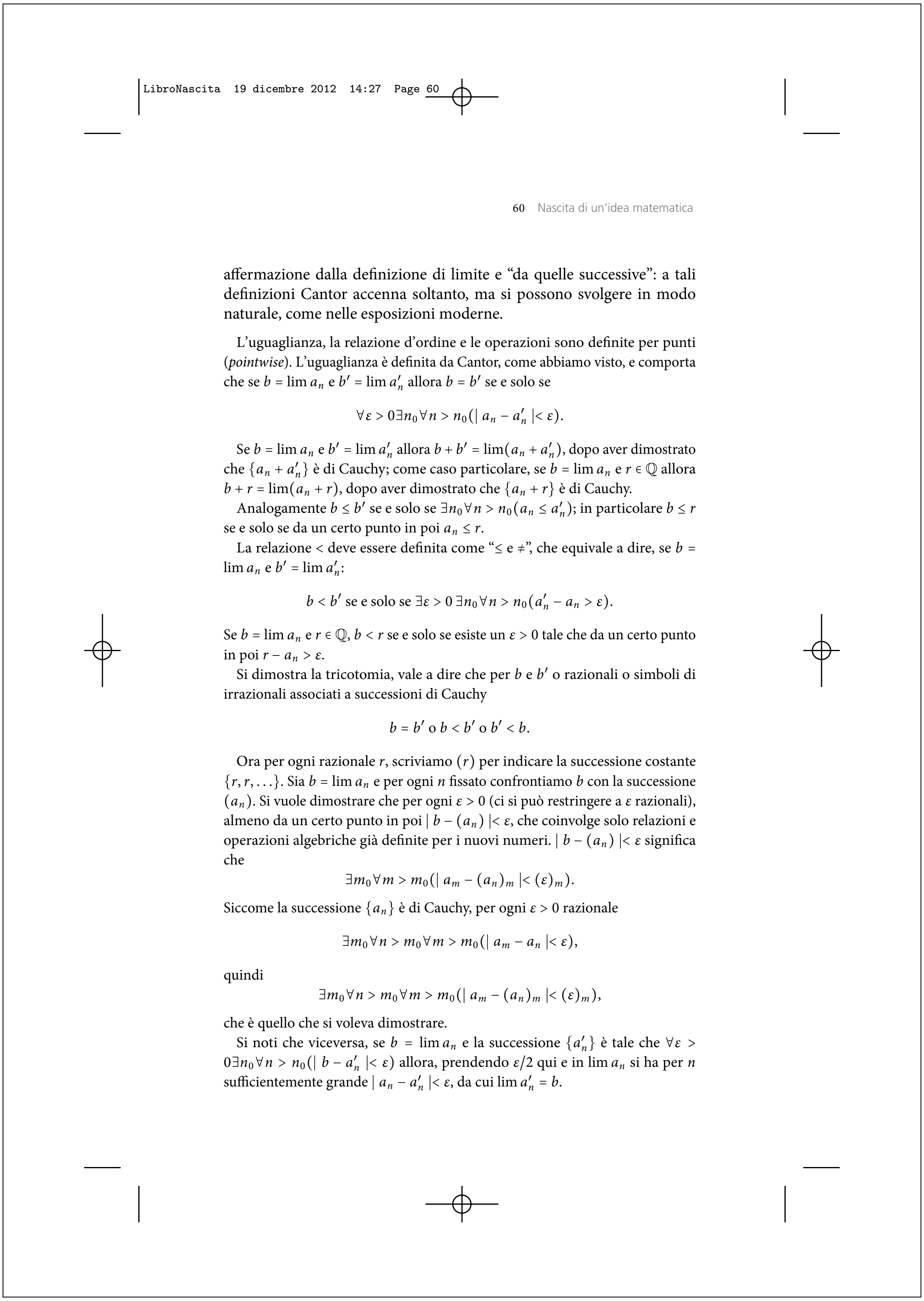
Die ☾ 30 decembris 1566.

4 *ante* possunt *del.* pãt *A*

[70] lege Boetium: ⋄ Fabrum in musicis elementis.:

[r]

Figure 20: More diagrams from the critical edition of Francesco Maurolico's *Musica*.



LibroNascita 19 dicembre 2012 14:27 Page 60

60 Nascita di un'idea matematica

affermazione dalla definizione di limite e "da quelle successive": a tali definizioni Cantor accenna soltanto, ma si possono svolgere in modo naturale, come nelle esposizioni moderne.

L'uguaglianza, la relazione d'ordine e le operazioni sono definite per punti (*pointwise*). L'uguaglianza è definita da Cantor, come abbiamo visto, e comporta che se $b = \lim a_n$ e $b' = \lim a'_n$ allora $b = b'$ se e solo se

$$\forall \varepsilon > 0 \exists n_0 \forall n > n_0 (| a_n - a'_n | < \varepsilon).$$

Se $b = \lim a_n$ e $b' = \lim a'_n$ allora $b + b' = \lim(a_n + a'_n)$, dopo aver dimostrato che $\{a_n + a'_n\}$ è di Cauchy; come caso particolare, se $b = \lim a_n$ e $r \in \mathbb{Q}$ allora $b + r = \lim(a_n + r)$, dopo aver dimostrato che $\{a_n + r\}$ è di Cauchy.

Analogamente $b \leq b'$ se e solo se $\exists n_0 \forall n > n_0 (a_n \leq a'_n)$; in particolare $b \leq r$ se e solo se da un certo punto in poi $a_n \leq r$.

La relazione $<$ deve essere definita come "$\leq$ e $\neq$", che equivale a dire, se $b = \lim a_n$ e $b' = \lim a'_n$:

$$b < b' \text{ se e solo se } \exists \varepsilon > 0\, \exists n_0 \forall n > n_0 (a'_n - a_n > \varepsilon).$$

Se $b = \lim a_n$ e $r \in \mathbb{Q}$, $b < r$ se e solo se esiste un $\varepsilon > 0$ tale che da un certo punto in poi $r - a_n > \varepsilon$.

Si dimostra la tricotomia, vale a dire che per b e b' o razionali o simboli di irrazionali associati a successioni di Cauchy

$$b = b' \text{ o } b < b' \text{ o } b' < b.$$

Ora per ogni razionale r, scriviamo (r) per indicare la successione costante $\{r, r, \ldots\}$. Sia $b = \lim a_n$ e per ogni n fissato confrontiamo b con la successione (a_n). Si vuole dimostrare che per ogni $\varepsilon > 0$ (ci si può restringere a ε razionali), almeno da un certo punto in poi $| b - (a_n) | < \varepsilon$, che coinvolge solo relazioni e operazioni algebriche già definite per i nuovi numeri. $| b - (a_n) | < \varepsilon$ significa che

$$\exists m_0 \forall m > m_0 (| a_m - (a_n)_m | < (\varepsilon)_m).$$

Siccome la successione $\{a_n\}$ è di Cauchy, per ogni $\varepsilon > 0$ razionale

$$\exists m_0 \forall n > m_0 \forall m > m_0 (| a_m - a_n | < \varepsilon),$$

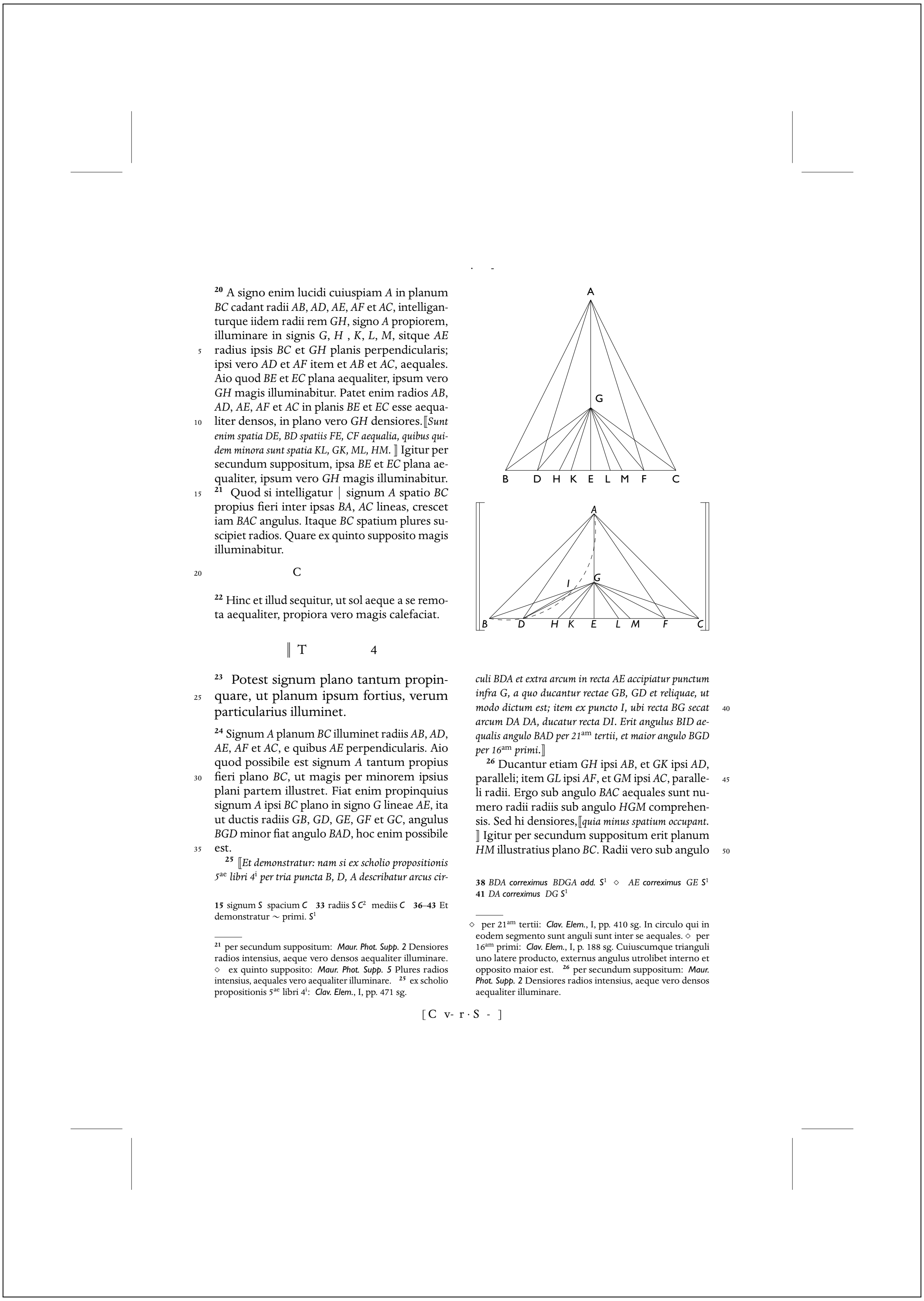
quindi

$$\exists m_0 \forall n > m_0 \forall m > m_0 (| a_m - (a_n)_m | < (\varepsilon)_m),$$

che è quello che si voleva dimostrare.

Si noti che viceversa, se $b = \lim a_n$ e la successione $\{a'_n\}$ è tale che $\forall \varepsilon > 0 \exists n_0 \forall n > n_0 (| b - a'_n | < \varepsilon)$ allora, prendendo $\varepsilon/2$ qui e in $\lim a_n$ si ha per n sufficientemente grande $| a_n - a'_n | < \varepsilon$, da cui $\lim a'_n = b$.

Figure 21: A page from a book on the development of mathematical logic.



20 A signo enim lucidi cuiuspiam *A* in planum *BC* cadant radii *AB*, *AD*, *AE*, *AF* et *AC*, intelliganturque iidem radii rem *GH*, signo *A* propiorem, illuminare in signis *G*, *H* , *K*, *L*, *M*, sitque *AE* radius ipsis *BC* et *GH* planis perpendicularis; ipsi vero *AD* et *AF* item et *AB* et *AC*, aequales. Aio quod *BE* et *EC* plana aequaliter, ipsum vero *GH* magis illuminabitur. Patet enim radios *AB*, *AD*, *AE*, *AF* et *AC* in planis *BE* et *EC* esse aequaliter densos, in plano vero *GH* densiores.⟦*Sunt enim spatia DE, BD spatiis FE, CF aequalia, quibus quidem minora sunt spatia KL, GK, ML, HM.* ⟧ Igitur per secundum suppositum, ipsa *BE* et *EC* plana aequaliter, ipsum vero *GH* magis illuminabitur.

21 Quod si intelligatur | signum *A* spatio *BC* propius fieri inter ipsas *BA*, *AC* lineas, crescet iam *BAC* angulus. Itaque *BC* spatium plures suscipiet radios. Quare ex quinto supposito magis illuminabitur.

C

22 Hinc et illud sequitur, ut sol aeque a se remota aequaliter, propiora vero magis calefaciat.

‖ T 4

23 Potest signum plano tantum propinquare, ut planum ipsum fortius, verum particularius illuminet.

24 Signum *A* planum *BC* illuminet radiis *AB*, *AD*, *AE*, *AF* et *AC*, e quibus *AE* perpendicularis. Aio quod possibile est signum *A* tantum propius fieri plano *BC*, ut magis per minorem ipsius plani partem illustret. Fiat enim propinquius signum *A* ipsi *BC* plano in signo *G* lineae *AE*, ita ut ductis radiis *GB*, *GD*, *GE*, *GF* et *GC*, angulus *BGD* minor fiat angulo *BAD*, hoc enim possibile est.

25 ⟦*Et demonstratur: nam si ex scholio propositionis 5*^{ae} *libri 4*^{i} *per tria puncta B, D, A describatur arcus circuli BDA et extra arcum in recta AE accipiatur punctum infra G, a quo ducantur rectae GB, GD et reliquae, ut modo dictum est; item ex puncto I, ubi recta BG secat arcum DA DA, ducatur recta DI. Erit angulus BID aequalis angulo BAD per 21*^{am} *tertii, et maior angulo BGD per 16*^{am} *primi.*⟧

26 Ducantur etiam *GH* ipsi *AB*, et *GK* ipsi *AD*, paralleli; item *GL* ipsi *AF*, et *GM* ipsi *AC*, paralleli radii. Ergo sub angulo *BAC* aequales sunt numero radii radiis sub angulo *HGM* comprehensis. Sed hi densiores,⟦*quia minus spatium occupant.* ⟧ Igitur per secundum suppositum erit planum *HM* illustratius plano *BC*. Radii vero sub angulo

15 signum *S* spacium *C* **33** radiis *S* C^2 mediis *C* **36–43** Et demonstratur ∼ primi. S^1

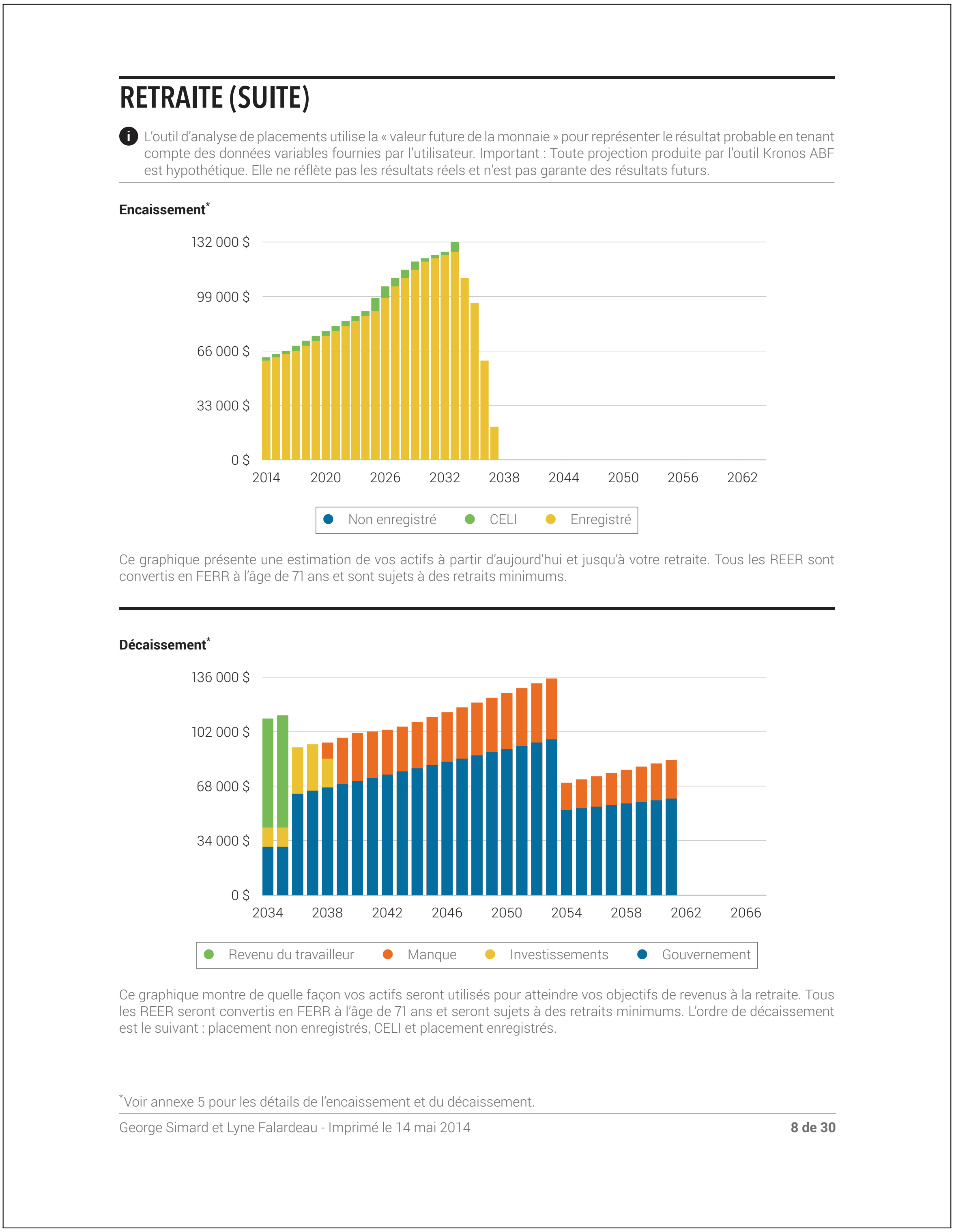
38 *BDA correximus BDGA add.* S^1 ◇ *AE correximus GE* S^1 **41** *DA correximus DG* S^1

[21] per secundum suppositum: *Maur. Phot. Supp.* 2 Densiores radios intensius, aeque vero densos aequaliter illuminare. ◇ ex quinto supposito: *Maur. Phot. Supp.* 5 Plures radios intensius, aequales vero aequaliter illuminare. [25] ex scholio propositionis 5^{ae} libri 4^{i}: *Clav. Elem.*, I, pp. 471 sg.

◇ per 21^{am} tertii: *Clav. Elem.*, I, pp. 410 sg. In circulo qui in eodem segmento sunt anguli sunt inter se aequales. ◇ per 16^{am} primi: *Clav. Elem.*, I, p. 188 sg. Cuiuscumque trianguli uno latere producto, externus angulus utrolibet interno et opposito maior est. [26] per secundum suppositum: *Maur. Phot. Supp.* 2 Densiores radios intensius, aeque vero densos aequaliter illuminare.

[C v- r · S -]

Figure 22: Geometric diagrams from the critical edition of Francesco Maurolico's *Optica*.



RETRAITE (SUITE)

ⓘ L'outil d'analyse de placements utilise la « valeur future de la monnaie » pour représenter le résultat probable en tenant compte des données variables fournies par l'utilisateur. Important : Toute projection produite par l'outil Kronos ABF est hypothétique. Elle ne reflète pas les résultats réels et n'est pas garante des résultats futurs.

Encaissement[*]

Ce graphique présente une estimation de vos actifs à partir d'aujourd'hui et jusqu'à votre retraite. Tous les REER sont convertis en FERR à l'âge de 71 ans et sont sujets à des retraits minimums.

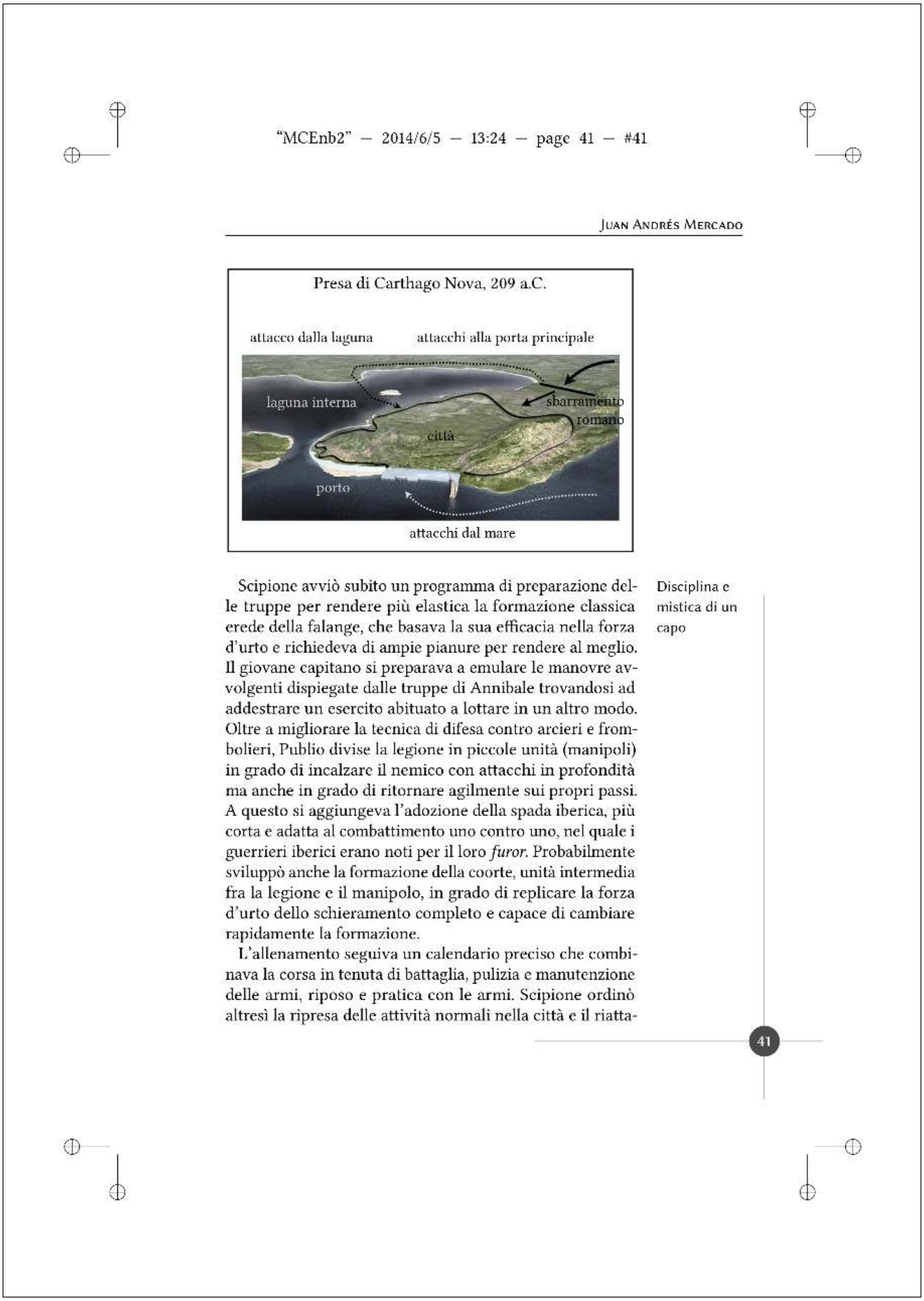
Décaissement[*]

Ce graphique montre de quelle façon vos actifs seront utilisés pour atteindre vos objectifs de revenus à la retraite. Tous les REER seront convertis en FERR à l'âge de 71 ans et seront sujets à des retraits minimums. L'ordre de décaissement est le suivant : placement non enregistrés, CELI et placement enregistrés.

[*] Voir annexe 5 pour les détails de l'encaissement et du décaissement.

George Simard et Lyne Falardeau - Imprimé le 14 mai 2014 **8 de 30**

Figure 23: Graphics from a financial report.



"MCEnb2" — 2014/6/5 — 13:24 — page 41 — #41

JUAN ANDRÉS MERCADO

Presa di Carthago Nova, 209 a.C.

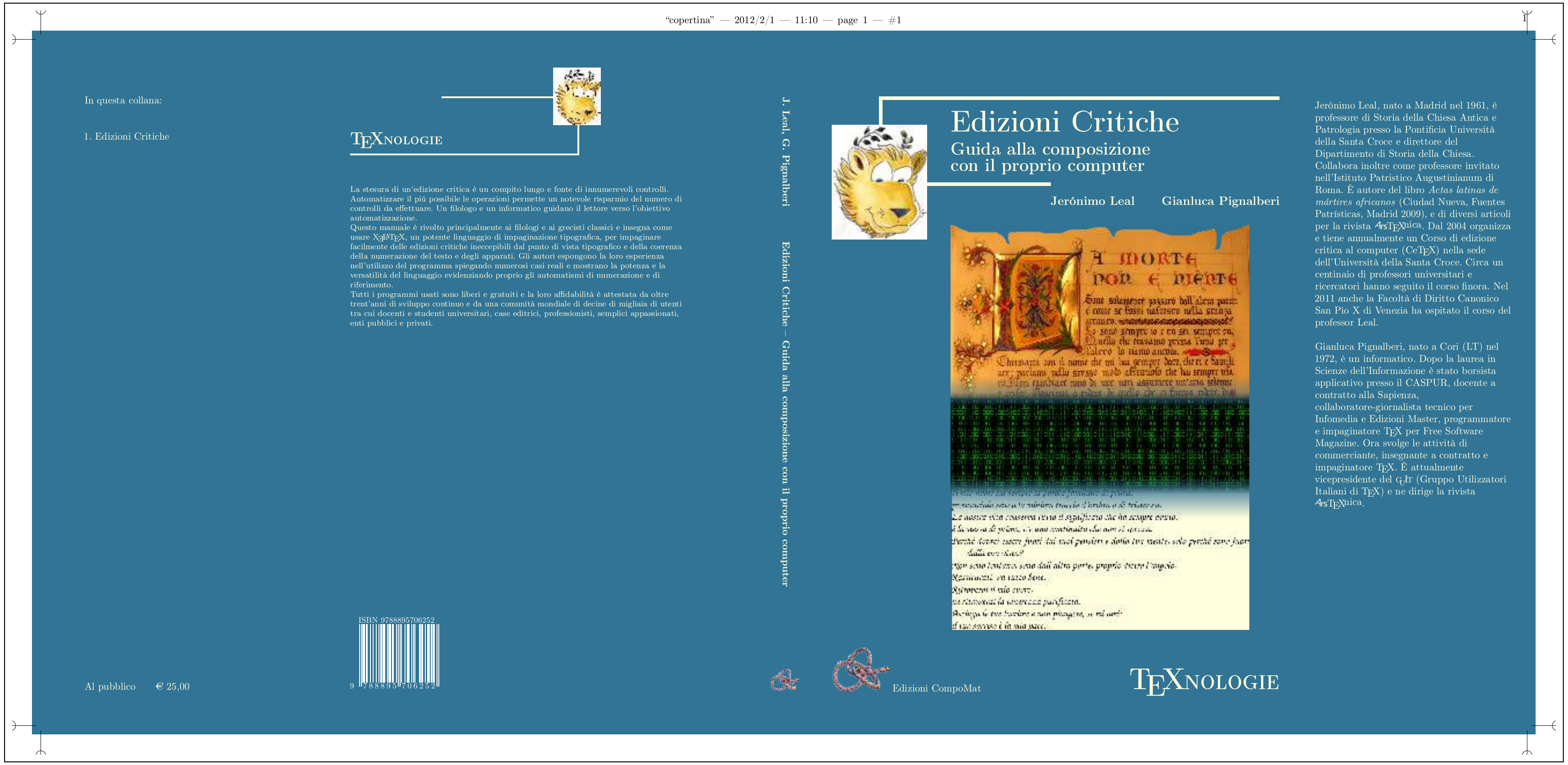
Disciplina e mistica di un capo

Scipione avviò subito un programma di preparazione delle truppe per rendere più elastica la formazione classica erede della falange, che basava la sua efficacia nella forza d'urto e richiedeva di ampie pianure per rendere al meglio. Il giovane capitano si preparava a emulare le manovre avvolgenti dispiegate dalle truppe di Annibale trovandosi ad addestrare un esercito abituato a lottare in un altro modo. Oltre a migliorare la tecnica di difesa contro arcieri e frombolieri, Publio divise la legione in piccole unità (manipoli) in grado di incalzare il nemico con attacchi in profondità ma anche in grado di ritornare agilmente sui propri passi. A questo si aggiungeva l'adozione della spada iberica, più corta e adatta al combattimento uno contro uno, nel quale i guerrieri iberici erano noti per il loro *furor*. Probabilmente sviluppò anche la formazione della coorte, unità intermedia fra la legione e il manipolo, in grado di replicare la forza d'urto dello schieramento completo e capace di cambiare rapidamente la formazione.

L'allenamento seguiva un calendario preciso che combinava la corsa in tenuta di battaglia, pulizia e manutenzione delle armi, riposo e pratica con le armi. Scipione ordinò altresì la ripresa delle attività normali nella città e il riatta-

41

FIGURE 24: A page from an EDUSC series.



"copertina" — 2012/2/1 — 11:10 — page 1 — #1

In questa collana:

1. Edizioni Critiche

TeXnologie

La stesura di un'edizione critica è un compito lungo e fonte di innumerevoli controlli. Automatizzare il piú possibile le operazioni permette un notevole risparmio del numero di controlli da effettuare. Un filologo e un informatico guidano il lettore verso l'obiettivo automatizzazione.
Questo manuale è rivolto principalmente ai filologi e ai grecisti classici e insegna come usare XeLaTeX, un potente linguaggio di impaginazione tipografica, per impaginare facilmente delle edizioni critiche ineccepibili dal punto di vista tipografico e della coerenza della numerazione del testo e degli apparati. Gli autori espongono la loro esperienza nell'utilizzo del programma spiegando numerosi casi reali e mostrano la potenza e la versatilità del linguaggio evidenziando proprio gli automatismi di numerazione e di riferimento.
Tutti i programmi usati sono liberi e gratuiti e la loro affidabilità è attestata da oltre trent'anni di sviluppo continuo e da una comunità mondiale di decine di migliaia di utenti tra cui docenti e studenti universitari, case editrici, professionisti, semplici appassionati, enti pubblici e privati.

ISBN 9788895706252

Al pubblico € 25,00

J. Leal, G. Pignalberi Edizioni Critiche – Guida alla composizione con il proprio computer

Edizioni Critiche

Guida alla composizione con il proprio computer

Jerónimo Leal Gianluca Pignalberi

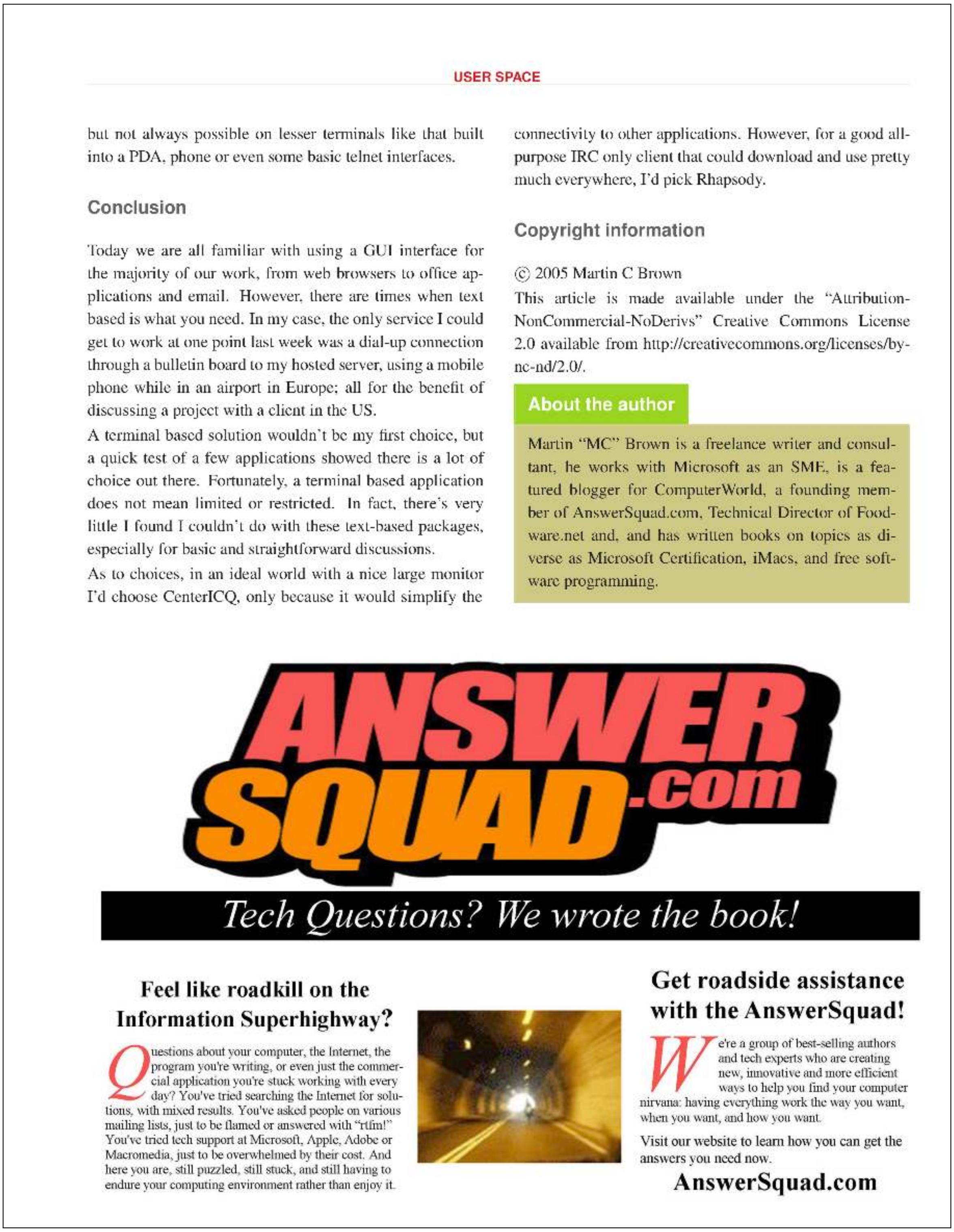
Edizioni CompoMat

TeXnologie

Jerónimo Leal, nato a Madrid nel 1961, è professore di Storia della Chiesa Antica e Patrologia presso la Pontificia Università della Santa Croce e direttore del Dipartimento di Storia della Chiesa. Collabora inoltre come professore invitato nell'Istituto Patristico Augustinianum di Roma. È autore del libro *Actas latinas de mártires africanos* (Ciudad Nueva, Fuentes Patrísticas, Madrid 2009), e di diversi articoli per la rivista ArsTeXnica. Dal 2004 organizza e tiene annualmente un Corso di edizione critica al computer (CeTeX) nella sede dell'Università della Santa Croce. Circa un centinaio di professori universitari e ricercatori hanno seguito il corso finora. Nel 2011 anche la Facoltà di Diritto Canonico San Pio X di Venezia ha ospitato il corso del professor Leal.

Gianluca Pignalberi, nato a Cori (LT) nel 1972, è un informatico. Dopo la laurea in Scienze dell'Informazione è stato borsista applicativo presso il CASPUR, docente a contratto alla Sapienza, collaboratore-giornalista tecnico per Infomedia e Edizioni Master, programmatore e impaginatore TeX per Free Software Magazine. Ora svolge le attività di commerciante, insegnante a contratto e impaginatore TeX. È attualmente vicepresidente del GuIt (Gruppo Utilizzatori Italiani di TeX) e ne dirige la rivista ArsTeXnica.

Figure 25: The dust cover jacket of one of the authors' book.



USER SPACE

but not always possible on lesser terminals like that built into a PDA, phone or even some basic telnet interfaces.

Conclusion

Today we are all familiar with using a GUI interface for the majority of our work, from web browsers to office applications and email. However, there are times when text based is what you need. In my case, the only service I could get to work at one point last week was a dial-up connection through a bulletin board to my hosted server, using a mobile phone while in an airport in Europe; all for the benefit of discussing a project with a client in the US.

A terminal based solution wouldn't be my first choice, but a quick test of a few applications showed there is a lot of choice out there. Fortunately, a terminal based application does not mean limited or restricted. In fact, there's very little I found I couldn't do with these text-based packages, especially for basic and straightforward discussions.

As to choices, in an ideal world with a nice large monitor I'd choose CenterICQ, only because it would simplify the connectivity to other applications. However, for a good all-purpose IRC only client that could download and use pretty much everywhere, I'd pick Rhapsody.

Copyright information

© 2005 Martin C Brown

This article is made available under the "Attribution-NonCommercial-NoDerivs" Creative Commons License 2.0 available from http://creativecommons.org/licenses/by-nc-nd/2.0/.

About the author

Martin "MC" Brown is a freelance writer and consultant, he works with Microsoft as an SME, is a featured blogger for ComputerWorld, a founding member of AnswerSquad.com, Technical Director of Foodware.net and, and has written books on topics as diverse as Microsoft Certification, iMacs, and free software programming.

ANSWER SQUAD .com

Tech Questions? We wrote the book!

Feel like roadkill on the Information Superhighway?

Questions about your computer, the Internet, the program you're writing, or even just the commercial application you're stuck working with every day? You've tried searching the Internet for solutions, with mixed results. You've asked people on various mailing lists, just to be flamed or answered with "rtfm!" You've tried tech support at Microsoft, Apple, Adobe or Macromedia, just to be overwhelmed by their cost. And here you are, still puzzled, still stuck, and still having to endure your computing environment rather than enjoy it.

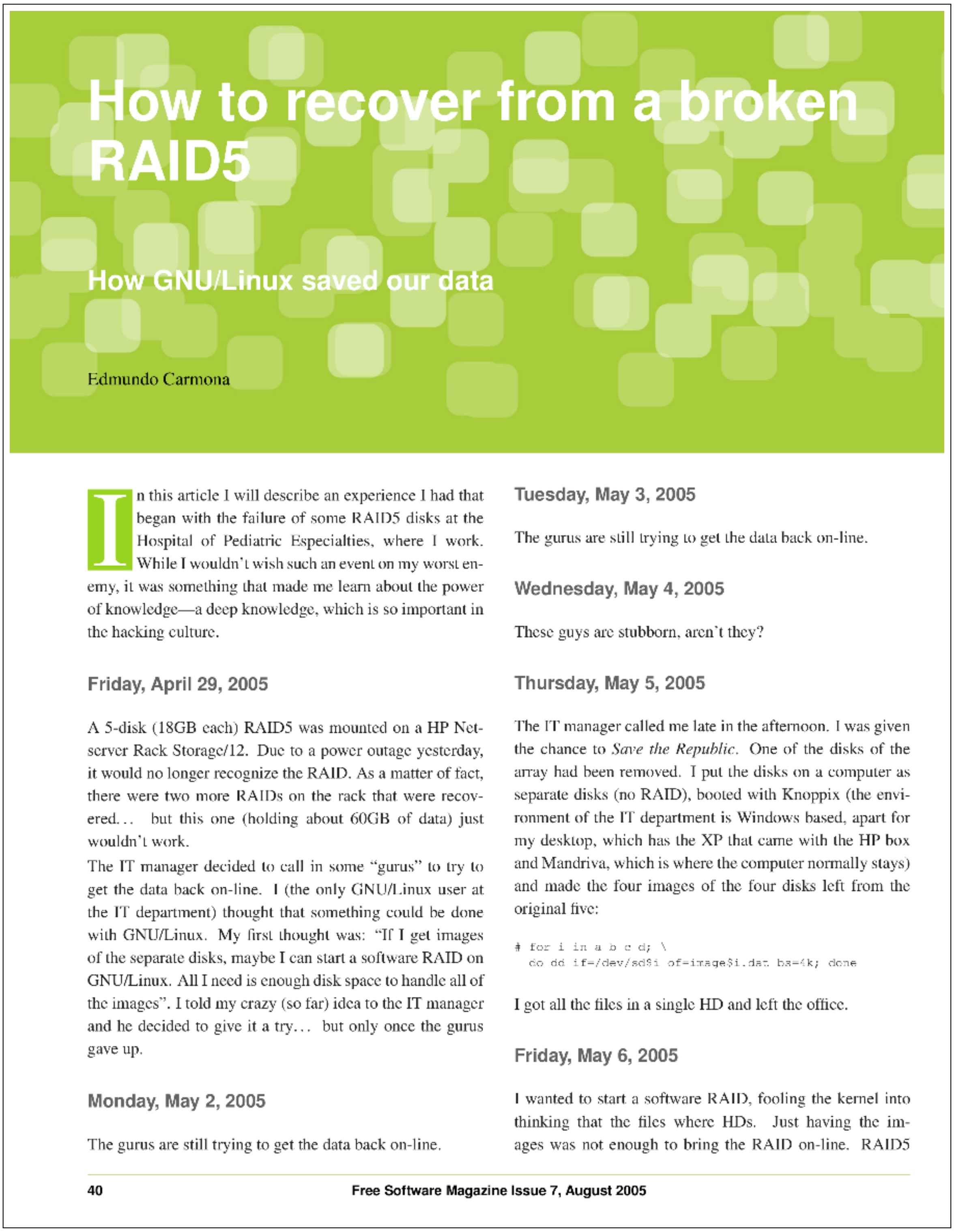
Get roadside assistance with the AnswerSquad!

We're a group of best-selling authors and tech experts who are creating new, innovative and more efficient ways to help you find your computer nirvana: having everything work the way you want, when you want, and how you want.

Visit our website to learn how you can get the answers you need now.

AnswerSquad.com

Figure 26: One page from Free Software Magazine n. 7 (camera ready for Lulu.com).



How to recover from a broken RAID5

How GNU/Linux saved our data

Edmundo Carmona

In this article I will describe an experience I had that began with the failure of some RAID5 disks at the Hospital of Pediatric Especialties, where I work. While I wouldn't wish such an event on my worst enemy, it was something that made me learn about the power of knowledge—a deep knowledge, which is so important in the hacking culture.

Friday, April 29, 2005

A 5-disk (18GB each) RAID5 was mounted on a HP Netserver Rack Storage/12. Due to a power outage yesterday, it would no longer recognize the RAID. As a matter of fact, there were two more RAIDs on the rack that were recovered… but this one (holding about 60GB of data) just wouldn't work.

The IT manager decided to call in some "gurus" to try to get the data back on-line. I (the only GNU/Linux user at the IT department) thought that something could be done with GNU/Linux. My first thought was: "If I get images of the separate disks, maybe I can start a software RAID on GNU/Linux. All I need is enough disk space to handle all of the images". I told my crazy (so far) idea to the IT manager and he decided to give it a try… but only once the gurus gave up.

Monday, May 2, 2005

The gurus are still trying to get the data back on-line.

Tuesday, May 3, 2005

The gurus are still trying to get the data back on-line.

Wednesday, May 4, 2005

These guys are stubborn, aren't they?

Thursday, May 5, 2005

The IT manager called me late in the afternoon. I was given the chance to *Save the Republic*. One of the disks of the array had been removed. I put the disks on a computer as separate disks (no RAID), booted with Knoppix (the environment of the IT department is Windows based, apart for my desktop, which has the XP that came with the HP box and Mandriva, which is where the computer normally stays) and made the four images of the four disks left from the original five:

```
# for i in a b c d; \
  do dd if=/dev/sd$i of=image$i.dat bs=4k; done
```

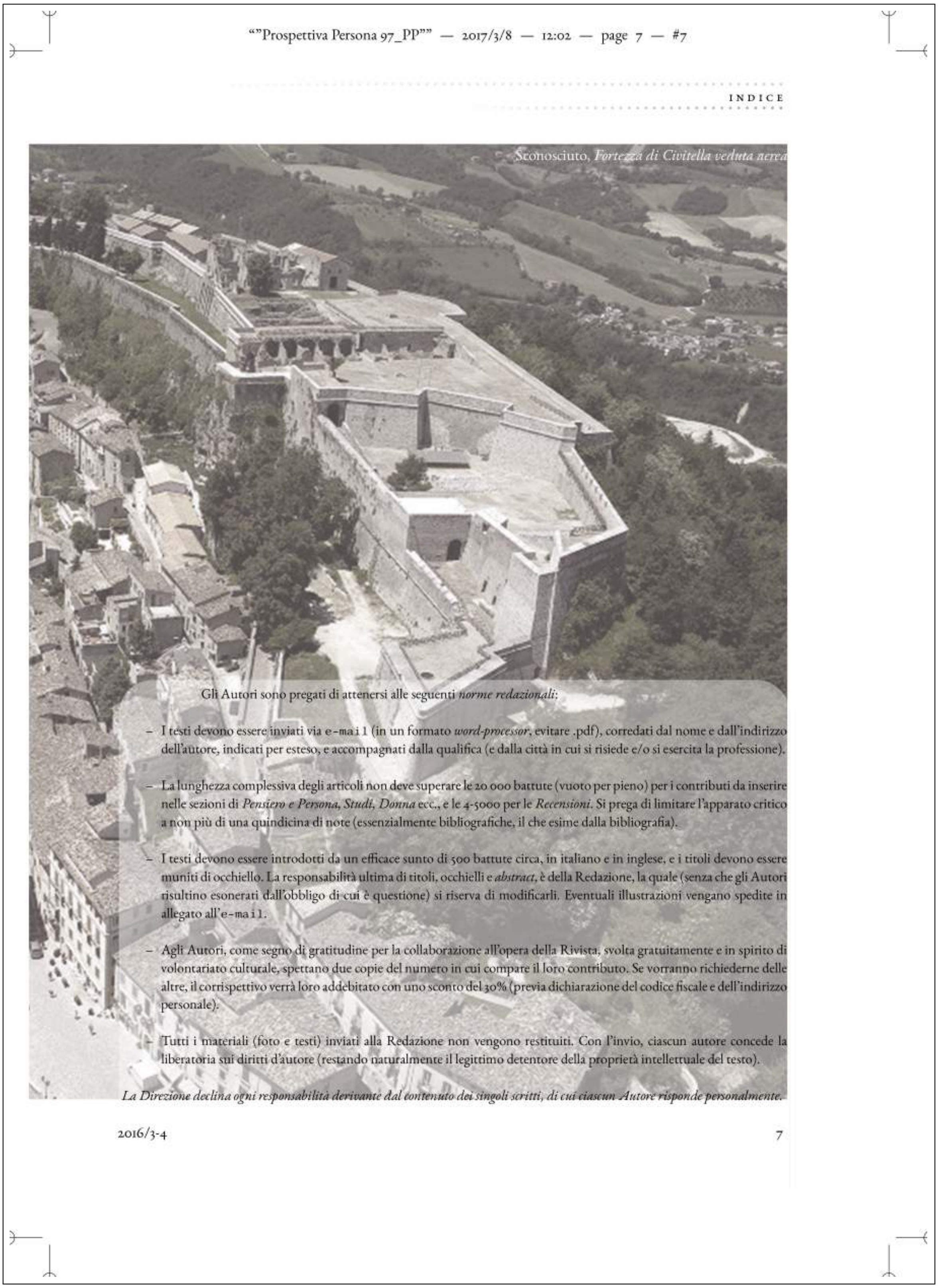
I got all the files in a single HD and left the office.

Friday, May 6, 2005

I wanted to start a software RAID, fooling the kernel into thinking that the files where HDs. Just having the images was not enough to bring the RAID on-line. RAID5

40　　**Free Software Magazine Issue 7, August 2005**

Figure 27: Another page from Free Software Magazine n. 7.



“”Prospettiva Persona 97_PP”” — 2017/3/8 — 12:02 — page 7 — #7

INDICE

Sconosciuto, *Fortezza di Civitella veduta aerea*

Gli Autori sono pregati di attenersi alle seguenti *norme redazionali*:

- I testi devono essere inviati via `e-mail` (in un formato *word-processor*, evitare .pdf), corredati dal nome e dall’indirizzo dell’autore, indicati per esteso, e accompagnati dalla qualifica (e dalla città in cui si risiede e/o si esercita la professione).

- La lunghezza complessiva degli articoli non deve superare le 20 000 battute (vuoto per pieno) per i contributi da inserire nelle sezioni di *Pensiero e Persona, Studi, Donna* ecc., e le 4-5000 per le *Recensioni*. Si prega di limitare l’apparato critico a non più di una quindicina di note (essenzialmente bibliografiche, il che esime dalla bibliografia).

- I testi devono essere introdotti da un efficace sunto di 500 battute circa, in italiano e in inglese, e i titoli devono essere muniti di occhiello. La responsabilità ultima di titoli, occhielli e *abstract*, è della Redazione, la quale (senza che gli Autori risultino esonerati dall’obbligo di cui è questione) si riserva di modificarli. Eventuali illustrazioni vengano spedite in allegato all’`e-mail`.

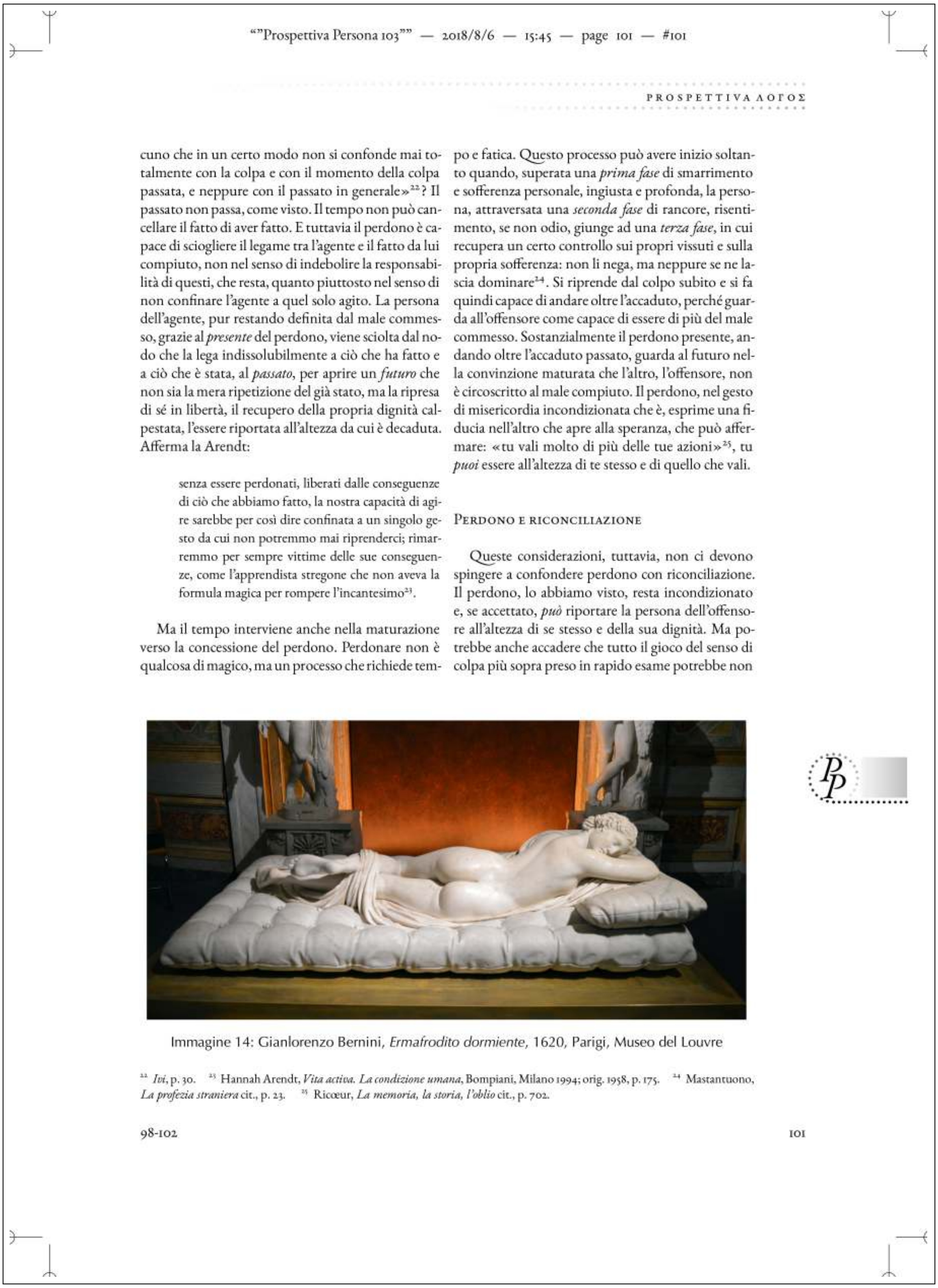
- Agli Autori, come segno di gratitudine per la collaborazione all’opera della Rivista, svolta gratuitamente e in spirito di volontariato culturale, spettano due copie del numero in cui compare il loro contributo. Se vorranno richiederne delle altre, il corrispettivo verrà loro addebitato con uno sconto del 30% (previa dichiarazione del codice fiscale e dell’indirizzo personale).

- Tutti i materiali (foto e testi) inviati alla Redazione non vengono restituiti. Con l’invio, ciascun autore concede la liberatoria sui diritti d’autore (restando naturalmente il legittimo detentore della proprietà intellettuale del testo).

La Direzione declina ogni responsabilità derivante dal contenuto dei singoli scritti, di cui ciascun Autore risponde personalmente.

2016/3-4 7

Figure 28: Prospettiva Persona editorial rules.



“”Prospettiva Persona 103”” — 2018/8/6 — 15:45 — page 101 — #101

PROSPETTIVA ΛΟΓΟΣ

cuno che in un certo modo non si confonde mai totalmente con la colpa e con il momento della colpa passata, e neppure con il passato in generale»[22]? Il passato non passa, come visto. Il tempo non può cancellare il fatto di aver fatto. E tuttavia il perdono è capace di sciogliere il legame tra l'agente e il fatto da lui compiuto, non nel senso di indebolire la responsabilità di questi, che resta, quanto piuttosto nel senso di non confinare l'agente a quel solo agito. La persona dell'agente, pur restando definita dal male commesso, grazie al *presente* del perdono, viene sciolta dal nodo che la lega indissolubilmente a ciò che ha fatto e a ciò che è stata, al *passato*, per aprire un *futuro* che non sia la mera ripetizione del già stato, ma la ripresa di sé in libertà, il recupero della propria dignità calpestata, l'essere riportata all'altezza da cui è decaduta. Afferma la Arendt:

> senza essere perdonati, liberati dalle conseguenze di ciò che abbiamo fatto, la nostra capacità di agire sarebbe per così dire confinata a un singolo gesto da cui non potremmo mai riprenderci; rimarremmo per sempre vittime delle sue conseguenze, come l'apprendista stregone che non aveva la formula magica per rompere l'incantesimo[23].

Ma il tempo interviene anche nella maturazione verso la concessione del perdono. Perdonare non è qualcosa di magico, ma un processo che richiede tempo e fatica. Questo processo può avere inizio soltanto quando, superata una *prima fase* di smarrimento e sofferenza personale, ingiusta e profonda, la persona, attraversata una *seconda fase* di rancore, risentimento, se non odio, giunge ad una *terza fase*, in cui recupera un certo controllo sui propri vissuti e sulla propria sofferenza: non li nega, ma neppure se ne lascia dominare[24]. Si riprende dal colpo subito e si fa quindi capace di andare oltre l'accaduto, perché guarda all'offensore come capace di essere di più del male commesso. Sostanzialmente il perdono presente, andando oltre l'accaduto passato, guarda al futuro nella convinzione maturata che l'altro, l'offensore, non è circoscritto al male compiuto. Il perdono, nel gesto di misericordia incondizionata che è, esprime una fiducia nell'altro che apre alla speranza, che può affermare: «tu vali molto di più delle tue azioni»[25], tu *puoi* essere all'altezza di te stesso e di quello che vali.

Perdono e riconciliazione

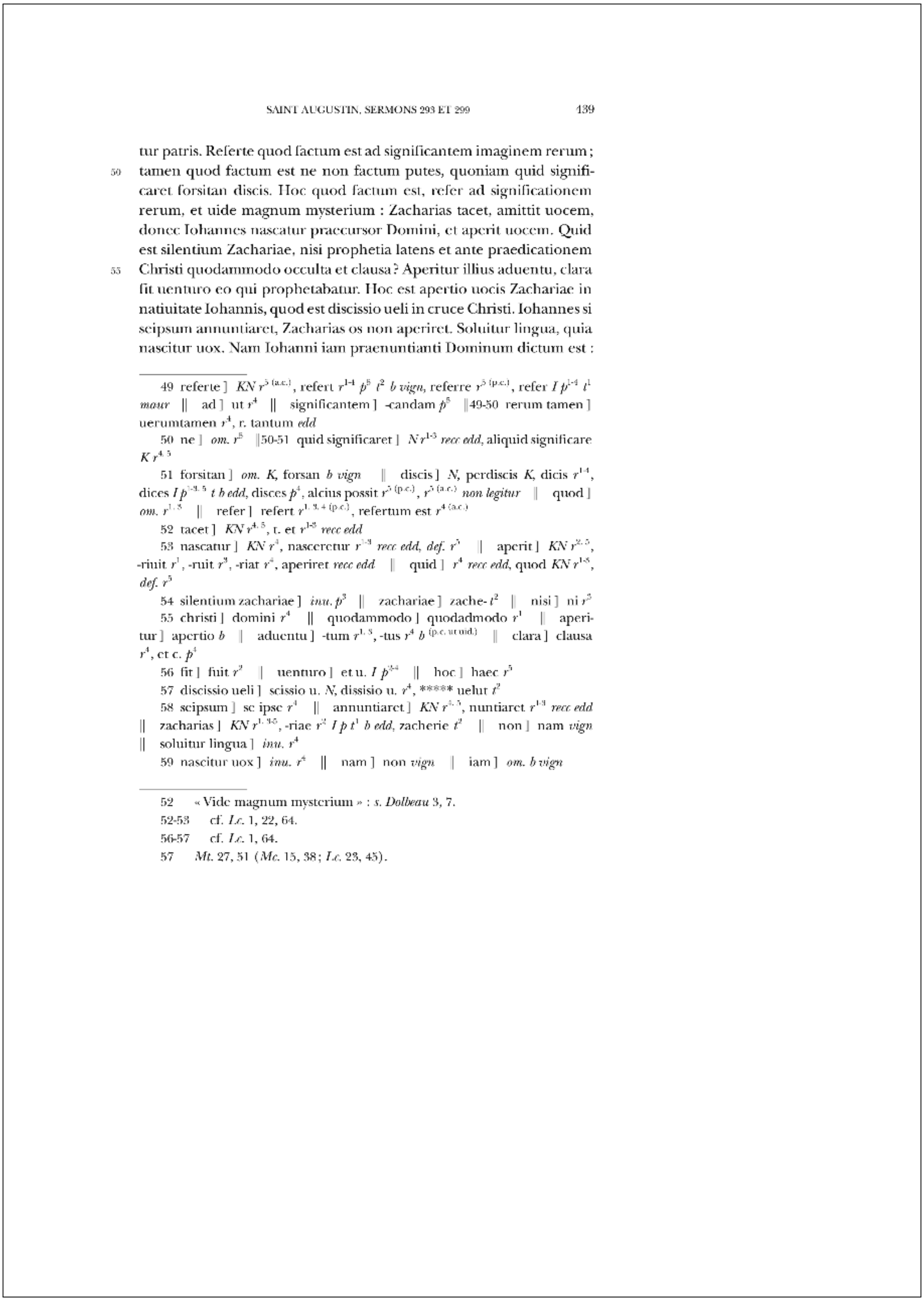
Queste considerazioni, tuttavia, non ci devono spingere a confondere perdono con riconciliazione. Il perdono, lo abbiamo visto, resta incondizionato e, se accettato, *può* riportare la persona dell'offensore all'altezza di se stesso e della sua dignità. Ma potrebbe anche accadere che tutto il gioco del senso di colpa più sopra preso in rapido esame potrebbe non

Immagine 14: Gianlorenzo Bernini, *Ermafrodito dormiente*, 1620, Parigi, Museo del Louvre

[22] *Ivi*, p. 30. [23] Hannah Arendt, *Vita activa. La condizione umana*, Bompiani, Milano 1994; orig. 1958, p. 175. [24] Mastantuono, *La profezia straniera* cit., p. 23. [25] Ricœur, *La memoria, la storia, l'oblio* cit., p. 702.

98-102 101

Figure 29: A page from the journal Prospettiva Persona.



SAINT AUGUSTIN, SERMONS 293 ET 299 139

tur patris. Referte quod factum est ad significantem imaginem rerum ; tamen quod factum est ne non factum putes, quoniam quid significaret forsitan discis. Hoc quod factum est, refer ad significationem rerum, et uide magnum mysterium : Zacharias tacet, amittit uocem, donec Iohannes nascatur praecursor Domini, et aperit uocem. Quid est silentium Zachariae, nisi prophetia latens et ante praedicationem Christi quodammodo occulta et clausa ? Aperitur illius aduentu, clara fit uenturo eo qui prophetabatur. Hoc est apertio uocis Zachariae in natiuitate Iohannis, quod est discissio ueli in cruce Christi. Iohannes si scipsum annuntiaret, Zacharias os non aperiret. Soluitur lingua, quia nascitur uox. Nam Iohanni iam praenuntianti Dominum dictum est :

49 referte] *KN* $r^{5\,(a.c.)}$, refert r^{1-4} p^5 t^2 *b vign*, referre $r^{5\,(p.c.)}$, refer *I* p^{1-4} t^1 *maur* || ad] ut r^4 || significantem] -candam p^5 ||49-50 rerum tamen] uerumtamen r^4, r. tantum *edd*

50 ne] *om.* r^5 ||50-51 quid significaret] *N* r^{1-3} *recc edd*, aliquid significare *K* $r^{4,5}$

51 forsitan] *om. K*, forsan *b vign* || discis] *N*, perdiscis *K*, dicis r^{1-4}, dices *I* $p^{1-3,5}$ *t b edd*, disces p^4, alcius possit $r^{5\,(p.c.)}$, $r^{5\,(a.c.)}$ *non legitur* || quod] *om.* $r^{1,3}$ || refer] refert $r^{1,3,4\,(p.c.)}$, refertum est $r^{4\,(a.c.)}$

52 tacet] *KN* $r^{4,5}$, t. et r^{1-3} *recc edd*

53 nascatur] *KN* r^4, nasceretur r^{1-3} *recc edd, def.* r^5 | aperit] *KN* $r^{2,5}$, -riuit r^1, -ruit r^3, -riat r^4, aperiret *recc edd* || quid] r^4 *recc edd*, quod *KN* r^{1-3}, *def.* r^5

54 silentium zachariae] *inu.* p^3 || zachariae] zache- t^2 || nisi] ni r^3

55 christi] domini r^4 || quodammodo] quodadmodo r^1 || aperitur] apertio *b* | aduentu] -tum $r^{1,3}$, -tus r^4 *b* $^{(p.c. ut uid.)}$ || clara] clausa r^4, et c. p^4

56 fit] fuit r^2 | uenturo] et u. *I* p^{2-4} || hoc] haec r^5

57 discissio ueli] scissio u. *N*, dissisio u. r^4, ***** uelut t^2

58 scipsum] se ipse r^4 || annuntiaret] *KN* $r^{4,5}$, nuntiaret r^{1-3} *recc edd* || zacharias] *KN* $r^{1,3-5}$, -riae r^2 *I p* t^1 *b edd*, zacherie t^2 || non] nam *vign* || soluitur lingua] *inu.* r^4

59 nascitur uox] *inu.* r^4 || nam] non *vign* | iam] *om. b vign*

52 « Vide magnum mysterium » : *s. Dolbeau* 3, 7.

52-53 cf. *Lc.* 1, 22, 64.

56-57 cf. *Lc.* 1, 64.

57 *Mt.* 27, 51 (*Mc.* 15, 38 ; *Lc.* 23, 45).

Figure 30: A François Dolbeau critical edition.

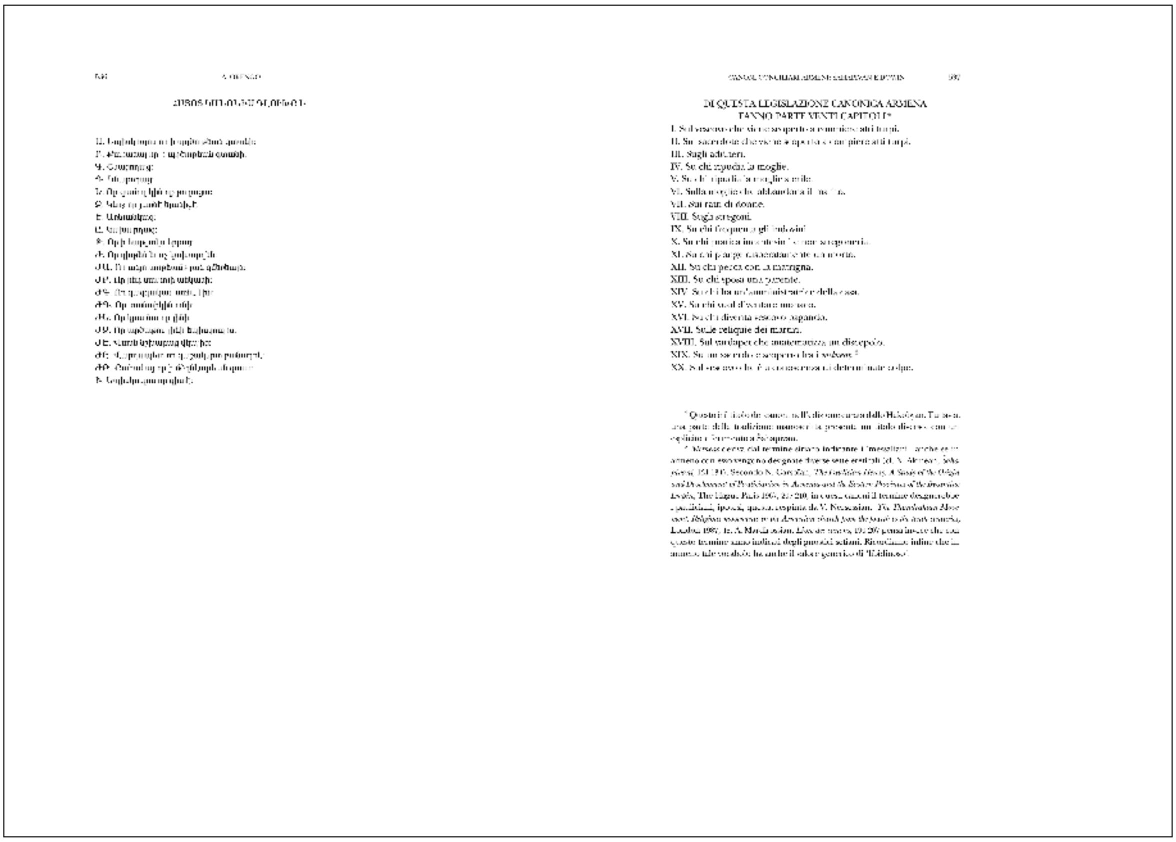


Figure 31: A parallel translation (Armenian-Italian) published in Augustinianum.

environment) with auto-completion, syntax highlighting, PDF visualization capabilities and much more functions.

Some other dedicated IDEs are Kile, TEX Works, TEXstudio, TEXnic Center. Users can configure them to decide what typesetting engine have to be used, to debug their documents and so on. The latest listed here, despite being a promising IDE, has been last updated back in 2014, which means that it has been dismissed.

Overleaf is a web site that allows users to create LATEX documents and collaborate on them. It offers an on-the-fly visualization of the resulting PDF.

The Wikipedia page Wikipedia (2019) lists a lot of editors and users can try one or more of them to pick their favorite. The most part of them are intended to work on the source code. They are better than a simple text editor because their capabilities and facilities can help users to be more focused and productive. Some of them are WYSIWYM (what you see is what you mean): despite the document looks so much different from the final document, it shows the structure in a way clearer for the authors and they can try to visualize the final document in their mind, before viewing as an on-screen preview.

The only real WYSIWYG editor seems to be TEXmacs, a GNU program inspired by Emacs and TEX, but totally unrelated to them. It does not even use TEX to typeset the final document.

In case you do not have a favorite editor (for instance, ours are vi and Emacs) you may try some of them and decide which one is yours.

In the next section we present some highlights on LYX.

10 LYX, the WYSIWYG (?) Editor that LATEXs Your Documents

First of all, this section title is not completely true: the right title should have been "LYX, the WYSI*s*WYG..." i.e., the only "what you see is *sometimes* what you get". The Wikipedia page Wikipedia (2019) states that LYX is properly a WYSIWYM editor.

Users who usually write their text documents with a WYSIWYG editor like Microsoft Word or LibreOffice Writer will find very hard to think of their documents the way LATEX requires. They feel comfortable, and possibly inspired, with an editor mimicking a white paper sheet. These users would probably be happy to use LYX, a program that looks very much like their favorite WP and that uses TEX as typesetting engine to get a LATEX document. They run LYX, see a nice user interface similar to those of the most famous WPs and... "Where is the white page?" Just a miserable yellowish band with a blinking cursor. Figure 32 shows LYX's new document. They do not need to won-

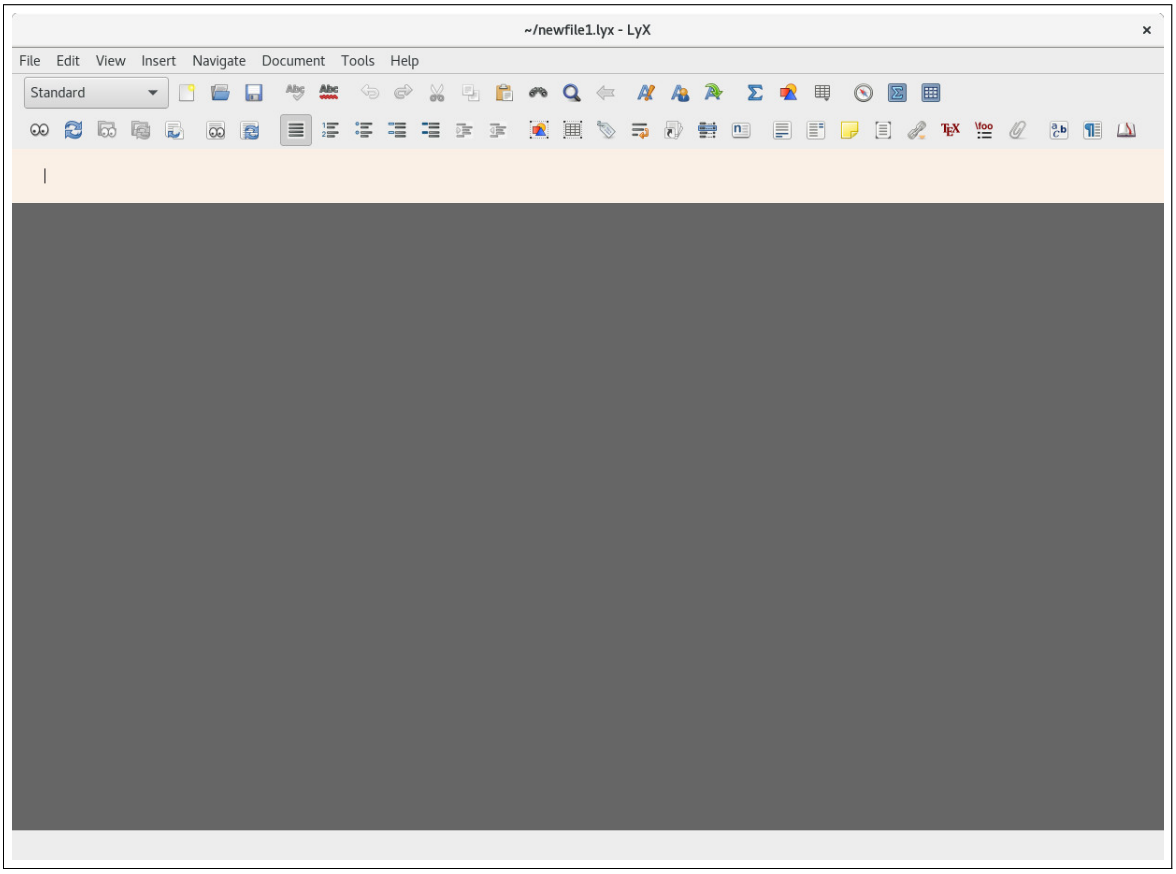
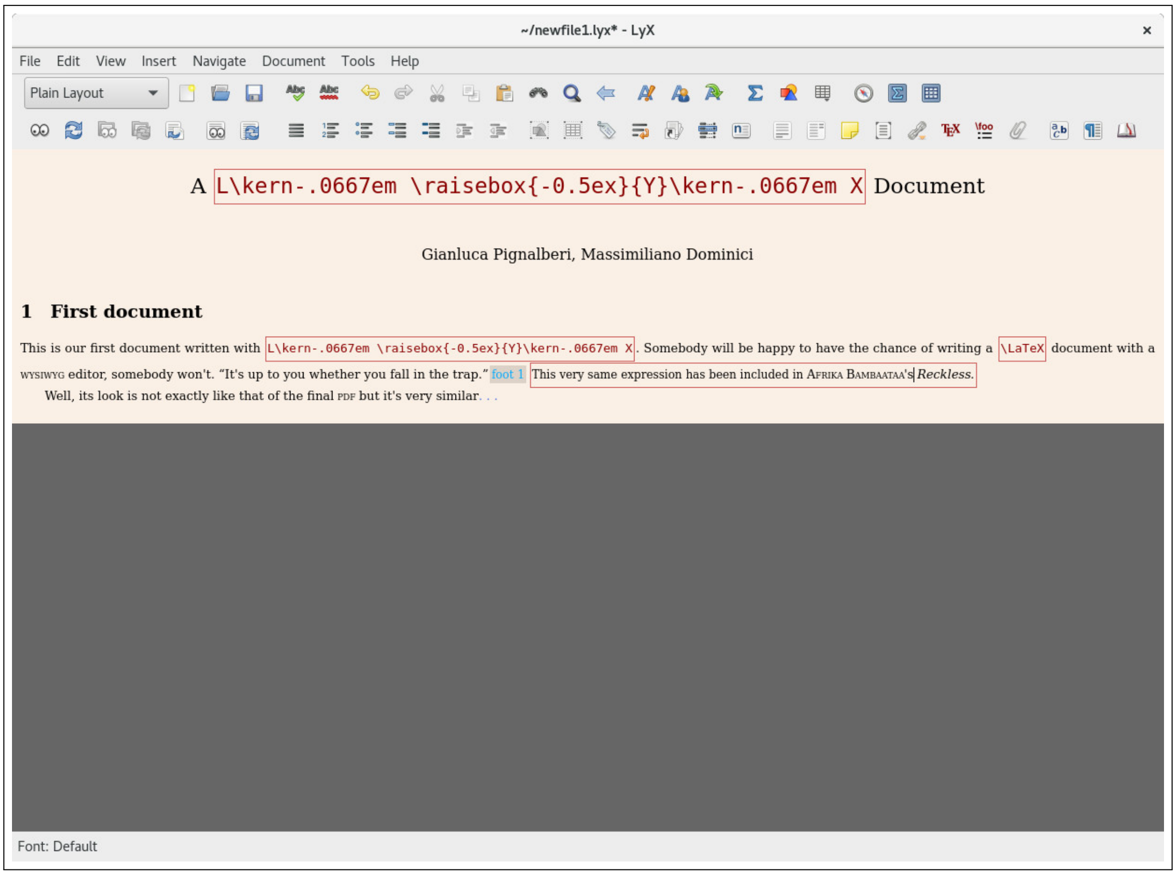


FIGURE 32: LYX and a new document.



A L\kern-.0667em \raisebox{-0.5ex}{Y}\kern-.0667em X Document

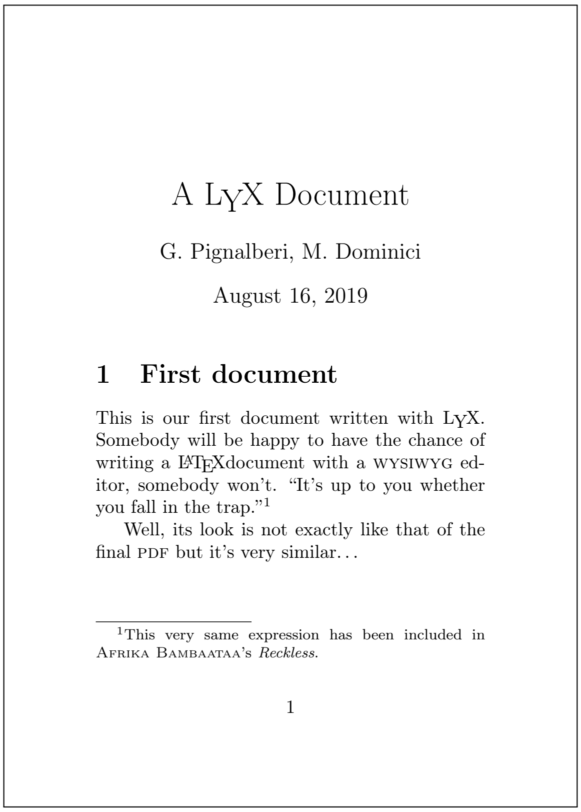
Gianluca Pignalberi, Massimiliano Dominici

1 First document

This is our first document written with L\kern-.0667em \raisebox{-0.5ex}{Y}\kern-.0667em X. Somebody will be happy to have the chance of writing a \LaTeX document with a WYSIWYG editor, somebody won't. "It's up to you whether you fall in the trap." foot 1 This very same expression has been included in AFRIKA BAMBAATAA's *Reckless*.

Well, its look is not exactly like that of the final PDF but it's very similar. . .

FIGURE 33: LYX and a new document.



A LyX Document

G. Pignalberi, M. Dominici

August 16, 2019

1 First document

This is our first document written with LyX. Somebody will be happy to have the chance of writing a LaTeXdocument with a WYSIWYG editor, somebody won't. "It's up to you whether you fall in the trap."[1]

Well, its look is not exactly like that of the final PDF but it's very similar...

[1] This very same expression has been included in Afrika Bambaataa's *Reckless.*

1

Figure 34: LyX and the new document exported in PDF.

der how will they write the whole text: the band enlarges as they keep writing.

A LyX file is not directly a LaTeX document, but it is very easy to export it in that format, so to post-produce it with LaTeX. Figure 33 shows that a document written into LyX only resembles the corresponding PDF (in figure 34), but the essence is there and that is the reason for Wikipedia (2019) to consider it WYSIWYM: we can inject TeX code into such a document to add unknown-to-LyX specific strings; we can "decide" with our mouse whether a portion of text is a title, a section, a footnote; we can apply emphasis or small caps (that the interface tags as "author" because some bibliography styles want the authors in small caps) to a portion of text highlighted with the mouse. LyX helps you manage a bibliography, lets you insert specific LaTeX features such as index entries, cross references and labels, tables of contents and many other elements.

In our basic example we used the default document class—article—with a A4 page size. Of course you can pick your needed type of document from Document→Settings... menu; you have a lot of things that you can customize there.

Click on the eyes to see the final PDF. When it suits your needs, you can export it with File→Export menu.

A Summary of textcomp commands

Apparently no manual groups textcomp commands all together.[16] We hope that our table 4, that shows all the textcomp symbols, helps you all in being more productive

You surely realized that two of the listed symbols are not visible. Those symbols, `\textascendercompwordmark` and `\textcapitalcompwordmark`, are "two additional compound word marks [...] that have the height of the ascender or capitals in the font, respectively." (Mittelbach and Goossens, 2013, p. 365). Those symbols are an extension of the LaTeX `\textcompwordmark`. They are zero-width characters (actually spaces) useful to prevent unwanted ligatures (do you remember the shelfful example? Try `shelf\textcompwordmark ful`) or to place an accent between two letters, as in the example of Mittelbach and Goossens (2013), the abbreviation of the German suffix -burg: `b\u\textcompwordmark g` and `B\u\textcapitalcompwordmark G`. In this case the zero-width symbol is the parameter of the accent command.

References

LaTeX 3 Project Team (2005). *LaTeX 2ε font selection.* Readable with `texdoc fntguide`.

Blanco, José Luis (2015). «Word or LaTeX typesetting: which one is more productive? Finally, scientifically assessed». Mapping Ignorance. https://mappingignorance.org/2015/04/06/word-or-latex-typesetting-which-one-is-more-productive-finally-scientifically-assessed/.

Bloch, Laurent (2017). «Efficacité comparée de latex et de ms-word». https://www.laurentbloch.net/MySpip3/Efficacite-comparee-de-LaTeX-et-de-MS-Word.

Burton, Tim (1997). *The Melancholy Death of Oyster Boy & Other Stories.* Rob Weisbach Books.

Carlisle, David, Scott Pakin and Alexander Holt (2001). *The Great, Big List of LaTeX Symbols.* https://www.rpi.edu/dept/arc/training/latex/LaTeX_symbols.pdf.

Charette, François (2015). *Polyglossia: An Alternative to Babel for XeLaTeX and LuaLaTeX.* Readable with `texdoc polyglossia`.

[16] Well, we found an old manual (Carlisle *et al.*, 2001) that alphabetically groups textcomp commands and symbols into table 18. Some of those symbols are currently dismissed and a lot more have been added since then. Mittelbach and Goossens (2013, pp. 362–368) explains in details textcomp and table 7.6 summarizes its symbols.

TABLE 4: textcomp commands and symbols. We present them in the very same order of appearance in textcomp.sty.

COMMAND	=	COMMAND	=	COMMAND	=
`\capitalcedilla`	¸	`\capitalogonek`	˛	`\capitalgrave`	`
`\capitalacute`	´	`\capitalcircumflex`	ˆ	`\capitaltilde`	˜
`\capitaldieresis`	¨	`\capitalhungarumlaut`	˝	`\capitalring`	˚
`\capitalcaron`	ˇ	`\capitalbreve`	˘	`\capitalmacron`	¯
`\capitaldotaccent`	˙	`\textcapitalcompwordmark`		`\textascendercompwordmark`	
`\textquotestraightbase`	'	`\textquotestraightdblbase`	"	`\texttwelveudash`	—
`\textthreequartersemdash`	—	`\textdollar`	$	`\textquotesingle`	'
`\textasteriskcentered`	∗	`\textfractionsolidus`	⁄	`\textminus`	−
`\textlbrackdbl`	⟦	`\textrbrackdbl`	⟧	`\textasciigrave`	`
`\texttildelow`	˷	`\textasciibreve`	˘	`\textasciicaron`	ˇ
`\textgravedbl`	˵	`\textacutedbl`	″	`\textdagger`	†
`\textdaggerdbl`	‡	`\textbardbl`	‖	`\textperthousand`	‰
`\textbullet`	•	`\textcelsius`	℃	`\textflorin`	ƒ
`\texttrademark`	™	`\textcent`	¢	`\textsterling`	£
`\textyen`	¥	`\textbrokenbar`	¦	`\textsection`	§
`\textasciidieresis`	¨	`\textcopyright`	©	`\textordfeminine`	ª
`\textlnot`	¬	`\textregistered`	®	`\textasciimacron`	¯
`\textdegree`	°	`\textpm`	±	`\texttwosuperior`	²
`\textthreesuperior`	³	`\textasciiacute`	´	`\textmu`	µ
`\textparagraph`	¶	`\textperiodcentered`	·	`\textonesuperior`	¹
`\textordmasculine`	º	`\textonequarter`	¼	`\textonehalf`	½
`\textthreequarters`	¾	`\texttimes`	×	`\textdiv`	÷
`\texteuro`	€	`\textohm`	Ω	`\textestimated`	℮
`\textcurrency`	¤	`\capitaltie`	⁀	`\newtie`	⁀
`\capitalnewtie`	⁀	`\textleftarrow`	←	`\textrightarrow`	→
`\textblank`	␢	`\textdblhyphen`	⹀	`\textzerooldstyle`	0
`\textoneoldstyle`	1	`\texttwooldstyle`	2	`\textthreeoldstyle`	3
`\textfouroldstyle`	4	`\textfiveoldstyle`	5	`\textsixoldstyle`	6
`\textsevenoldstyle`	7	`\texteightoldstyle`	8	`\textnineoldstyle`	9
`\textlangle`	〈	`\textrangle`	〉	`\textmho`	℧
`\textbigcircle`	◯	`\textuparrow`	↑	`\textdownarrow`	↓
`\textborn`	★	`\textdivorced`	⚮	`\textdied`	†
`\textleaf`	🍃	`\textmarried`	⚭	`\textmusicalnote`	♪
`\textdblhyphenchar`	=	`\textdollaroldstyle`	$	`\textcentoldstyle`	¢
`\textcolonmonetary`	₡	`\textwon`	₩	`\textnaira`	₦
`\textguarani`	₲	`\textpeso`	₱	`\textlira`	₤
`\textrecipe`	℞	`\textinterrobang`	‽	`\textinterrobangdown`	⸘
`\textdong`	₫	`\textpertenthousand`	‱	`\textpilcrow`	¶
`\textbaht`	฿	`\textnumero`	№	`\textdiscount`	⁒
`\textopenbullet`	◦	`\textservicemark`	℠	`\textlquill`	⁅
`\textrquill`	⁆	`\textcopyleft`	🄯	`\textcircledP`	℗
`\textreferencemark`	※	`\textsurd`	√	`\textcircled`	◯
		`\t`	⁀		

DAVID P. CARLISLE AND THE LATEX3 PROJECT (2017). *Packages in the 'graphics' bundle*. Readable with `texdoc graphicx`.

GIACOMELLI, Roberto and Gianluca PIGNALBERI (2018). «Typesetting and highlighting Unicode source code with LaTeX: a package comparison». *ArsTeXnica*, (18), pp. 39–54.

GREGORIO, Enrico (2010). «Installare TeX Live 2010 su Ubuntu». *ArsTeXnica*, (10), pp. 7–13.

KNAUFF, Markus and Jelica NEJASMIC (2014). «An efficiency comparison of document preparation systems used in academic research and development». *PLoS ONE*, **9** (12), p. e115 069. https://doi.org/10.1371/journal.pone.0115069.

KNUTH, Donald E. (1999). *Digital Typography*. Center for the Study of Language and Information, Stanford, CA.

KOPKA, Helmut and Patrick W. DALY (2004). *Guide to LaTeX*. Addison Wesley, Boston, 4th edition.

KÜHL, Philipp and Daniel KIRSCH (2019). «Detexify latex handwritten symbol recognition». http://detexify.kirelabs.org/classify.html.

LAMPORT, Leslie (1987). MakeIndex*: An Index Processor for LaTeX*. Readable with `texdoc makeindex`.

LAVAGNINO, John (2003). *The* endnotes *package*. Readable with `texdoc endnotes`.

MITTELBACH, Frank and Michel GOOSSENS (2013). *The LaTeX companion*. Addison Wesley, Boston, 2nd edition.

MITTELBACH, Frank, Robin FAIRBAIRNS, Werner LEMBERG and LATEX 3 PROJECT TEAM (2016). *LaTeX 2ε font encodings*. Readable with `texdoc encguide`.

OETIKER, Tobias, Hubert PARTL, Irene HYNA and Elisabeth SCHLEGL (2018). *The Not So Short Introduction to LaTeX 2ε*. A4 format, Version 6.2, February 28, 2018. Readable with `texdoc lshort`.

PAKIN, Scott (2017). *The Comprehensive LaTeX Symbol List*. Readable with `texdoc symbols-a4`.

POWERS, Shelley, Jerry PEEK, Tim O'REILLY and Mike LOUKIDES (2002). *Unix Power Tools*. O'Reilly, Sebastopol, CA.

ROBBINS, Arnold, Elbert HANNAH and Linda LAMB (2008). *Learning the vi and Vim Editors*. O'Reilly, Sebastopol, 7th edition.

ROBERTSON, Will and Khaled HOSNY (2017). *The* fontspec *package*. Readable with `texdoc fontspec`.

ROONEY, Garrett (2005). *Practical Subversion*. APress, Berkeley.

SCHMIDT, Walter (2006). «Font selection in LaTeX: The most frequently asked questions». *The PracTeX Journal*. https://www.tug.org/pracjourn/2006-1/schmidt/schmidt.pdf.

STACKEXCHANGE (2011). «macros - Different command definitions with and without optional argument». https://tex.stackexchange.com/questions/308/different-command-definitions-with-and-without-optional-argument.

— (2016). «macros - What di the commands inside the LaTeX logo do?» https://tex.stackexchange.com/questions/313527/what-do-the-commands-inside-the-latex-logo-do.

UMEKI, Hideo (2010). *The* geometry *package*. Readable with `texdoc geometry`.

WIKIPEDIA (2019). «Comparison of TeX editors». https://en.wikipedia.org/wiki/Comparison_of_TeX_editors.

Bonus Section: Solution

You cannot pretend to read it that easy. Get a mirror! This is the 500th anniversary of Leonardo da Vinci's death.

All of the documents shown in figures 15–31 have been typeset by Massimiliano Dominici or Gianluca Pignalberi with LaTeX, being it pdfLaTeX, XeLaTeX or LuaLaTeX.

▷ Gianluca Pignalberi
`g dot pignalberi at gmail dot com`

▷ Massimiliano Dominici
`mlgdominici at gmail dot com`

TEX, LATEX and math

Enrico Gregorio

Abstract

We discuss some aspects of mathematical typesetting: choice of symbols, code abstraction, fine details. Relationships between math typesetting and international standards are examined. A final section on typesetting of numbers and units reports on some recent developments in the field.

Sommario

Si discutono alcuni aspetti della tipografia matematica: scelta dei simboli, astrazione del codice, dettagli più fini. Si esaminano anche connessioni tra la tipografia matematica e gli standard internazionali. La sezione conclusiva tocca il problema della composizione di numeri e unità di misura alla luce di sviluppi recenti.

1 Introduction

We all know that TEX was born out of Knuth's discomfort after having seen the proofs of the new edition of the first volume of his *magnum opus* "The Art of Computer Programming".

Many papers have been written by Knuth himself and by others on the topic of math typesetting. Here I'd like to present some personal ideas on the subject, coming from almost thirty year long experience in mathematical typesetting. I'll also present some recent developments and new tricks made available with expl3.

2 A very short lead-in to math in TEX

Every TEX guru knows that TEX is always in one of three *modes*:

- horizontal mode,
- vertical mode,
- math mode.

Actually, there are circumstances when TEX is in no mode at all (when writing to external files, for the curious).

Each mode comes into two flavors, but we're interested only in math mode. Knuth calls the two flavors 'math mode' and 'display math mode'. In order to better distinguish between them, I'll call the former 'inline math mode', so the unadorned 'math mode' will denote both.

There are subtle, well, not so much so, differences between the two flavors; beginners are most impressed by $\sum_{k=1}^{n} k^2 = \frac{1}{3}n\left(n+\frac{1}{2}\right)(n+1)$ that suddenly becomes

$$\sum_{k=1}^{n} k^2 = \frac{1}{3}n\left(n+\frac{1}{2}\right)(n+1)$$

when displayed and a very common question is 'how do I get the limits above and below the summation symbol and real fractions, not that smallish replacement symbol?'

I've been a beginner myself; I discovered `\limits` and abused it. *Penitenziagite*, would have said Salvatore in "Il nome della rosa". Now I'm no longer a beginner and *know* why `\limits` should not be used, not to talk about the dreaded `\displaystyle` that sometimes is suggested to newbies. The proper way is just `\sum`.

To the contrary, beginners are usually much less impressed by the wrong typesetting in

$$A\backslash B = \{x|x \in A, x \notin B\}$$

but they are likely to shrug and move on, if they ever note it. Sometimes they see something's wrong and 'fix' the vertical bar by using `\,|\,` that's still wrong. Why is it wrong? The spacing is too small, of course, but there's more into the problem: two appearances of such a construction in the same document is a sin similar to what I describe to young basketball referees: "whoever calls a double foul during their career has called one too many". The correct answer is: first of all define a macro for the object, for instance,

```
\newcommand{\suchthat}{\,|\,}
```

(I'm talking LATEX, plain TEX users can translate). In case one asks, if `a+b` appears twice or more in a document there's no need to make a macro out of it; the separator in the set builder notation is a single conceptual object and so it *must* be typed by a single command.

About the spacing, one should realize that the reverse bar is a binary operation symbol and the vertical bar is a relation symbol. Both are already defined in all flavors of TEX and they are, respectively, `\setminus` and `\mid`, but it's still convenient and logically sound to define `\suchthat`, because `\mid` is a 'generic' name:

```
\newcommand{\suchthat}{\mid}
...
A \setminus B=\{x \suchthat
  x\in A, x\notin B\}
```

will typeset as

$$A \setminus B = \{x \mid x \in A, x \notin B\}$$

This is the version with the thin spaces

$$A \setminus B = \{x \,|\, x \in A, x \notin B\}$$

Compare closely the spaces around the vertical bar.

I'm not saying the last realization should be rejected as awfully wrong: personal judgment is always welcome when typography is concerned, after having studied the alternatives and common practice. Above all, consistency throughout a document is a must. I had to edit a paper where the separator was a bar or a colon or a semicolon, depending on which of the three authors had typed the formula. Defining `\suchthat` allows for delaying any decision about what symbol to use until the last minute. More on set builder notation later.

The TEXbook lists several symbol names, some have semantics attached to them, like `\setminus`, others don't, like `\mid` or `\otimes`.[1] Why is that? Some symbols have essentially a single use case, others appear in different branches of mathematics with different meanings. Everybody loves `\lhd` and `\unlhd`, right? The symbols typeset as $\lhd$ and $\unlhd$ respectively. I believe to have seen once what the names should suggest, but I forgot it. The symbols are common in group theory, where they denote 'normal subgroup': it's heartily recommended to group theorists to define a meaningful command for them. Oh, I was almost forgetting! Those are not considered as relation symbols, so a savvy group theorist will type in the document preamble

```
% normal subgroup
\newcommand{\ns}{\mathrel\lhd}
\newcommand{\nseq}{\mathrel\unlhd}
% subnormal subgroup
\newcommand{\sns}{\ns\ns}
```

The symbols are not among the core ones designed by Knuth; they first appeared in a symbol font distributed along with LATEX; possibly Lamport used them for his own papers as binary operators and the classification stuck. They were later included in amssymb.

What should an author do? The case of normal subgroups is clear: I surely wouldn't litter my paper with `\mathrel\lhd` each time I want to mention normal subgroups. However, suppose a paper frequently uses Euler's *totient function*, which has the well established tradition of being denoted by φ (the open version of phi). Is it better to use `\varphi` or to define `\euphi`? The latter. Upon receiving the proofs, the author realizes that *all* instances of `\varphi` print out ϕ, because the publisher uses a font that lacks the proper symbol. With `\euphi` it is a matter of doing a redefinition, probably borrowing the open phi from another font. We don't know when the instruction `\let\varphi=\phi` is performed, but using `\euphi` makes this irrelevant.

An important exception: in the abstract there should be *no* use of personal macros. It should be able to typeset with a 'naked' version of LATEX: it's very common nowadays that the abstract is fed to some web page that maybe uses MathML, MathJax or similar device for handing the text to browsers.

Going back to the normal subgroup symbol, one should know that every math symbol belongs to a class and there are seven of them:

- class 0, ordinary symbols;
- class 1, operators;
- class 2, binary operations;
- class 3, binary relations;
- class 4, opening symbols;
- class 5, closing symbols;
- class 6, punctuation.

TEX will set the spacing between symbols according to well defined rules. This is not the place to discuss them fully, see Gregorio (2009). Any object, as long as it is legal in math mode, can be defined to behave as if it belongs in one of the above classes by typing it as the argument to

```
\mathord \mathop \mathbin \mathrel
\mathopen \mathclose \mathpunct
```

For instance, the symbol for the determinant is internally carried out by something like

```
\mathop{\operator@font det}\nolimits
```

but there is a higher level interface available for declaring new symbols like this; for instance, one does

```
\DeclareMathOperator{\adj}{adj}
```

in order to introduce a symbol for the adjugate matrix. A one-shot operator can be input in the document by

```
\operatorname{adj}
```

The *-version of both commands makes for a symbol that carries limits above and below in display math mode, on the side when inline.

The unfortunately common perversion of denoting open intervals like $]a,b[$ needs input such as

```
\mathopen]a,b\mathclose[
```

[1] Generally LATEX kept the same names.

One can easily spot that something is wrong when just using `]a,b[` by looking at the difference between the two instances below

$$x \in]a,b[\qquad x \in \mathopen{]}a,b\mathclose{[}$$

In my calculus notes I type `\interval[o]{a,b}`. I can decide to be a perv by just changing a few lines in the definition. An open interval will be typeset as $(a \mathinner{\ldotp\ldotp} b)$, but I'm not bound in any way: I can go back to the comma again by just changing a line. Also, I like to write upper unbounded intervals like $(a \mathinner{\ldotp\ldotp} \to)$, but I use `\pinf` for the arrow, so I can make it to be typeset ∞ by acting on a single line, should I change my mind.

Upon entering math mode, TEX will construct a *math list* consisting of *math atoms*, each of which has a *nucleus*, a *subscript field* and a *superscript* field. When exiting from math mode, the math list will be transformed into a horizontal list according to the (complex) rules described in Appendix G of the TEXbook. These rules add spaces, as said before, but also take care of the bidimensionality of math formulas: superscripts, subscripts, fractions, accents, radicals, extensible delimiters and many more aspects.

Had Knuth been into theoretical physics, he probably would have added also "prescripts" for isotopes and staggered multiple subscripts and superscripts for tensors. Unfortunately he hasn't. See later for more on this topic.

3 Fine points of mathematics typing

The title is the same as chapter 18 in the TEXbook. Of course I won't go through Knuth's words. Since I'm talking LATEX and math, I assume that amsmath is loaded: no serious math typesetting can be done without it.

A point that's not touched upon in the TEXbook is 'when, really, consecutive equations should be aligned and where'. Browsing TEX.StackExchange reveals several examples of bad alignments.

A prominent example is a derivation of Cardano's formula[2] which I won't give the code for, but just three realizations that you can see in figure 1.

I often use the style "the good, the bad, and the ugly". There is actually an even uglier way, which is what the questioner was asking for, see figure 2.

What's the problem? The equals signs are not really related to one another. The *pairs of formulas* are related, the fact they are equations is almost irrelevant. Mixing ragged right and ragged left in one and the same paragraph (or display) makes for very hard reading. I'd instead be more generous with vertical spacing between the various braces and I have no doubt whatsoever that the leftmost realization is our Clint Eastwood. Look for holes in the typeset output and remove them.

Another example can be seen in figure 3.[3] You can judge by yourself what's the best way to present the display. My opinion is that the equals signs in the second column pair are not related to each other, so they're not to be aligned.

Linear systems are an exception, because their matrix-like structure is more important than holes. I recommend the wonderful systeme package by Ch. Tellechea (Tellechea, 2019). No doubt there are other exceptions: typography, and mathematical typography in particular, is a craft that doesn't obey to mechanical rules. A thin space may open up symbols and make them easier to read, adding a pair of parentheses may clear up an ambiguity, removing unnecessary parentheses may improve the quality of a formula. Compare top and bottom line

$$\begin{gathered}
a\frac{f(x+h)-f(x)}{h}+b\frac{g(x+h)-g(x)}{h}\\
a\,\frac{f(x+h)-f(x)}{h}+b\,\frac{g(x+h)-g(x)}{h}
\end{gathered}$$

and decide which one looks better. In my notes I used the bottom one when working the proof of linearity of the derivative. If I talk about "the function $g(z)=\sqrt{z-1}\,$", I add a thin space before ending inline math mode:

```
``the function $g(z)=\sqrt{z-1}\,$''
```

in order to avoid the clash between the vinculum and the quotes in "the function $g(z)=\sqrt{z-1}$". Try with a parenthesis after the radical to see another case: $(1+\sqrt{2})^{-1}$ versus $(1+\sqrt{2}\,)^{-1}$. In the latter case a thin space has been added.

Going to *very* fine details: does anybody notice the differences below? Consider the formulas

$$\log|x| \neq \log\lvert x\rvert \tag{7}$$

$$|\sin x| \neq \lvert\sin x\rvert \tag{8}$$

$$\|\operatorname{adj} A\| \neq \lVert\operatorname{adj} A\rVert \tag{9}$$

where the questionable typesetting is on the left. While the top left *could* be a typographic choice (so long as it is consistent), the other formulas in the left-hand sides are definitely wrong.

The mathtools package provides a very good facility for handling these cases, namely

```
\DeclarePairedDelimiter{\abs}{|}{|}
```

that allows to type `\abs{\sin x}` and forget about the dreaded thin space, which can also be avoided by

```
\lvert\sin x\rvert
```

[2] https://tex.stackexchange.com/questions/193581

[3] https://tex.stackexchange.com/questions/500472

$$
\begin{cases}
a^6+2a^3b^3+b^6=q^2\\
4a^3b^3=-\dfrac{4}{27}\,p^3
\end{cases}
\qquad
\left\{\begin{aligned}
a^6+2a^3b^3+b^6&=q^2\\
4a^3b^3&=-\frac{4}{27}\,p^3
\end{aligned}\right.
\qquad
\left\{\begin{aligned}
a^6+2a^3b^3+b^6&=q^2\\
4a^3b^3&=-\frac{4}{27}\,p^3
\end{aligned}\right.
$$

$$
\begin{cases}
a^3+b^3=-q\\
a^6-2a^3b^3+b^6=q^2+\dfrac{4}{27}\,p^3
\end{cases}
\qquad
\left\{\begin{aligned}
a^3+b^3&=-q\\
a^6-2a^3b^3+b^6&=q^2+\frac{4}{27}\,p^3
\end{aligned}\right.
\qquad
\left\{\begin{aligned}
a^3+b^3&=-q\\
a^6-2a^3b^3+b^6&=q^2+\frac{4}{27}\,p^3
\end{aligned}\right.
$$

$$
\begin{cases}
a^3+b^3=-q\\
(a^3-b^3)^2=q^2+\dfrac{4}{27}\,p^3
\end{cases}
\qquad
\left\{\begin{aligned}
a^3+b^3&=-q\\
(a^3-b^3)^2&=q^2+\frac{4}{27}\,p^3
\end{aligned}\right.
\qquad
\left\{\begin{aligned}
a^3+b^3&=-q\\
(a^3-b^3)^2&=q^2+\frac{4}{27}\,p^3
\end{aligned}\right.
$$

$$
\begin{cases}
a^3+b^3=-q\\
a^3-b^3=\sqrt{q^2+\dfrac{4}{27}\,p^3}
\end{cases}
\qquad
\left\{\begin{aligned}
a^3+b^3&=-q\\
a^3-b^3&=\sqrt{q^2+\frac{4}{27}\,p^3}
\end{aligned}\right.
\qquad
\left\{\begin{aligned}
a^3+b^3&=-q\\
a^3-b^3&=\sqrt{q^2+\frac{4}{27}\,p^3}
\end{aligned}\right.
$$

Figure 1: Three ways of laying out the derivation of Cardano's formula

$$
\left\{\begin{aligned}
a^6+2a^3b^3+b^6&=q^2\\
4a^3b^3&=-\frac{4}{27}\,p^3
\end{aligned}\right.
$$

$$
\left\{\begin{aligned}
a^3+b^3&=-q\\
a^6-2a^3b^3+b^6&=q^2+\frac{4}{27}\,p^3
\end{aligned}\right.
$$

$$
\left\{\begin{aligned}
a^3+b^3&=-q\\
(a^3-b^3)^2&=q^2+\frac{4}{27}\,p^3
\end{aligned}\right.
$$

$$
\left\{\begin{aligned}
a^3+b^3&=-q\\
a^3-b^3&=\sqrt{q^2+\frac{4}{27}\,p^3}
\end{aligned}\right.
$$

Figure 2: One of the worst alignment I can conceive

Which style to choose is a matter of personal preference and habit. I recommend not to abuse the facility: reserve it for *functions* such as absolute values, norms and similar objects. Don't exploit it for parenthesized expressions: something like

```
\paren{a+b}\paren{a-b}=a^2-b^2
```

hinders input reading and would print the same as `(a+b)(a-b)=a^2-b^2`. True, one could do

```
\paren[\big]{a\paren{(b+c}}
```

but is this really more legible than

```
\bigl( a (b + c) \bigr)
```

that keeps the usual mathematical structure? That is, assuming `\big` size is really necessary, which it isn't in the particular case.

Since I mentioned trigonometric functions, look at

$$\sqrt{\sin x}+\sqrt{\cos x}+\sqrt{\tan x}$$

and explain what's going wrong. Yes, the tittle makes the difference! It makes 'sin' higher than 'cos' and moves up the radical sign; similarly with 'tan'. In my trigonometry notes I have

```
\let\cos\undefined
\DeclareMathOperator{\cos}
  {cos\vphantom{i}}
\let\tan\undefined
\DeclareMathOperator{\tan}
  {tan\vphantom{i}}
```

with which the above formula would become

$$\sqrt{\sin x}+\sqrt{\cos x}+\sqrt{\tan x}$$

Radicals often need fine control in order to get them aligned with each other. Some appropriate trick involving `\vphantom` or `\smash` can fix things up:

$$\sqrt{x}+\sqrt{y}\neq\sqrt{x}+\sqrt{\smash[b]{y}}$$

Again, left is the questionable output; the formula on the right has been input as

```
\sqrt{x}+\sqrt{\smash[b]{y}}
```

The alternative

```
\sqrt{\mathstrut x}+\sqrt{\mathstrut y}
```

doesn't seem as attractive: $\sqrt{\mathstrut x}+\sqrt{\mathstrut y}$. Radicals would need a full chapter, so I'll stop here. One last thing: add a thin space when a radical is followed by a fence; similarly, add a thin space when a big operator (summation, product, integral) in display math mode is preceded by a fence and its limits are wide. Example

$$\Biggl(\sum_{k=1}^{n}a_k\Biggr)\neq\Biggl(\,\sum_{k=1}^{n}a_k\Biggr)$$

4 Upright or italic?

Rivers of (electronic) ink have been spilled trying to answer the question. Actually it cannot be answered: mathematicians and engineers agree to disagree. Physicists disagree with each other.

Part of the question is: should constants be typeset in upright font or not? The ISO 80000-2:2009 standard prescribes upright; adhering to this standard is mandatory in some technology and commercial fields. This is a good thing: people reading

$$
\begin{aligned}
\begin{pmatrix} A_\mu \\ \rho^*_\mu \end{pmatrix} &\to \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} A_\mu \\ \rho^*_\mu \end{pmatrix}, & \tan\theta &= \frac{g_{el}}{g_*} & (1)\\
\begin{pmatrix} \psi_L \\ \chi_L \end{pmatrix} &\to \begin{pmatrix} \cos\varphi_{\psi_L} & -\sin\varphi_{\psi_L} \\ \sin\varphi_{\psi_L} & \cos\varphi_{\psi_L} \end{pmatrix} \begin{pmatrix} \psi_L \\ \chi_L \end{pmatrix}, & \tan\varphi_{\psi_L} &= \frac{\Delta}{m} & (2)\\
\begin{pmatrix} \tilde\psi_R \\ \tilde\chi_R \end{pmatrix} &\to \begin{pmatrix} \cos\varphi_{\tilde\psi_R} & -\sin\varphi_{\tilde\psi_R} \\ \sin\varphi_{\tilde\psi_R} & \cos\varphi_{\tilde\psi_R} \end{pmatrix} \begin{pmatrix} \tilde\psi_R \\ \tilde\chi_R \end{pmatrix}, & \tan\varphi_{\tilde\psi_R} &= \frac{\tilde\Delta}{\tilde m} & (3)
\end{aligned}
$$

$$
\begin{aligned}
\begin{pmatrix} A_\mu \\ \rho^*_\mu \end{pmatrix} &\to \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} A_\mu \\ \rho^*_\mu \end{pmatrix}, & \tan\theta &= \frac{g_{el}}{g_*} & (4)\\
\begin{pmatrix} \psi_L \\ \chi_L \end{pmatrix} &\to \begin{pmatrix} \cos\varphi_{\psi_L} & -\sin\varphi_{\psi_L} \\ \sin\varphi_{\psi_L} & \cos\varphi_{\psi_L} \end{pmatrix} \begin{pmatrix} \psi_L \\ \chi_L \end{pmatrix}, & \tan\varphi_{\psi_L} &= \frac{\Delta}{m} & (5)\\
\begin{pmatrix} \tilde\psi_R \\ \tilde\chi_R \end{pmatrix} &\to \begin{pmatrix} \cos\varphi_{\tilde\psi_R} & -\sin\varphi_{\tilde\psi_R} \\ \sin\varphi_{\tilde\psi_R} & \cos\varphi_{\tilde\psi_R} \end{pmatrix} \begin{pmatrix} \tilde\psi_R \\ \tilde\chi_R \end{pmatrix}, & \tan\varphi_{\tilde\psi_R} &= \frac{\tilde\Delta}{\tilde m} & (6)
\end{aligned}
$$

FIGURE 3: Two similar alignments

a technical report or manual will have no doubt about the meaning of a symbol.[4] While I strongly disagree with several decisions of the ISO standard, on mathematical grounds, I accept the underlying philosophy towards uniformity in the technical fields. Surely I appreciate its ban on the mathematically wrong $\sin^{-1}$ and similar: the standard has disputable aspects, but it's never wrong from a mathematical point of view.

On the other hand, many mathematicians are traditionalists and prefer italics for constants such as e (the Euler number) and i (the imaginary unit). Euler and Gauss used italics for the latter, I'm among those who don't dare to challenge their authority. Of course, I know that mathematical notation has changed along time. I'd not use Cayley's original notation for matrices[5] because a better notation has developed. I follow the practice of setting standard function names in upright type (sine, cosine, logarithm and so on) even when ancient mathematicians didn't.

However, such decrees as 'symbols for vectors should be bold italic serif lowercase, for matrices should be bold italic serif uppercase, for tensors should be bold italic sans serif uppercase' make me smile: as a mathematician, I know that vectors, matrices and tensors are not different objects from a mathematical point of view. Matrices admit an easier two-dimensional representation: this is the 'big' difference.

For pedagogical reasons, I might use a distinctive typesetting for vectors and matrices in a students' textbook. In a research paper or graduate level book I'd probably not make any distinction, if not mandated by clarity. In this case I'd explain the notation choices at the beginning of the paper or book.

A very fine book by J. Dieudonné (DIEUDONNÉ, 1972), in the English edition by Academic Press, uses

- $\mathbf{R}$ or $\mathbf{C}$ for number sets,
- X for manifolds, E for vector bundles,
- A for vector space operators,
- $\mathrm{T}_x(\mathrm{X})$ or $\mathrm{T}_x(f)$ for the tangent space or linear mapping,
- $d_x\mathsf{f}$ or d_xf for the differential at x of a mapping (vector valued or scalar valued),
- $\boldsymbol{\mathsf{Z}}$ for tensor fields,

and several other conventions that are consistently followed across the book and the series. The book starts of with a nine page long notation section. The same notation is used in the original French version.

However, it happens that book translations use different conventions from the original. It is the case of W. Rudin's 'Real and Complex Analysis' (RUDIN, 1966) where the differential 'd' is in italics, whereas it's upright in the Italian translation published by Bollati-Boringhieri (RUDIN, 1974). I disagree with the publisher: maybe the editorial preference is for the upright 'd', but the author's style should be preserved as much as possible.

Not a big deal, one could think. No, this reflects on the *meaning* of the differential 'd'. There are several arguments in favor or against italics; my feeling is that most pure mathematicians prefer italics.

By the way, how to input the symbol in such a way that the convention can be changed at will? The simplest and more effective way is to define

```
\newcommand{\diff}{\mathop{}\!d}
```

[4] A problem with ISO standards is that they have to be *bought*; the one we're talking about prices 158 CHF, about 143 € or $160 at the current exchange rate.

[5] https://tex.stackexchange.com/q/487643

(or `\mathrm{d}` if one *really* prefers the abomination).

I believe to have learned it from Claudio Beccari through a `comp.text.tex` post. The code was credited to him in the paper Guiggiani and Mori (2008a),[6] but I'm not sure about the real source of this code pearl. Claudio Beccari had earlier proposed a much more complicated code (Beccari, 1997), namely

```
\makeatletter
\providecommand*{\diff}{%
  \@ifnextchar^{\DIfF}{\DIfF^{}}%
}
\makeatother
\def\DIfF^#1{%
  \mathop{\mathrm{\mathstrut d}}%
  \nolimits^{#1}%
  \gobblespace
}
\def\gobblespace{%
  \futurelet\diffarg\opspace
}
\def\opspace{%
  \let\DiffSpace\!%
  \ifx\diffarg(%
    \let\DiffSpace\relax
  \else
    \ifx\diffarg[%
      \let\DiffSpace\relax
    \else
      \ifx\diffarg\{%
        \let\DiffSpace\relax
      \fi
    \fi
  \fi
  \DiffSpace
}
```

What's the idea in the complicated definition? Look whether a superscript follows; if it doesn't, add a dummy one. Well, this is already wrong, because it adds `\scriptspace` unconditionally. After that, the next token is examined: if it is a fence, then don't add `\!`, because a `\mathop` is followed by a fence with no thin space; in case an ordinary symbol follows, the `\mathop` would add a thin space, which is removed by `\!`. Well, try it with `\diff\bigl(x+y\bigr)`. Next try the simpler definition and see! Where's the trick? The *empty* `\mathop` is followed by an ordinary symbol, the 'd'; we just need to remove the excess thin space! The thin space preceding the empty `\mathop` is inserted automatically by TeX following the rules. Thus we can define

```
\newcommand{\tder}[2]
  {\frac{\diff #1}{\diff #2}}
```

without worrying that spurious spaces may creep in. Instead

```
\iint\limits_{D} f(x,y) \diff x \diff y
```

will typeset as needed

$$\iint\limits_{D} f(x,y)\,dx\,dy$$

For differential forms

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

the spacing will be automatically right.

The same paper by Beccari (1997) proposes commands for the constants, namely

```
% The number ‘e’
\providecommand*{\eu}
  {\ensuremath{\mathrm{e}}}
% The imaginary unit
\providecommand*{\iu}
  {\ensuremath{\mathrm{j}}}
```

I strongly disagree with proposing `\ensuremath`; referring in the text to the Euler's number by

```
We use \eu\ to denote...
```

is by no means easier and clearer than

```
We use $\eu$ to denote...
```

One keystroke more? So what? That's a mathematical symbol so it ought to be typed in math mode, just like when we talk about the variable x. *My* definition would be

```
\newcommand{\eu}{\mathord{e}}
```

so typing `\eu` outside of math mode would raise an error. Change to `\mathrm` if you prefer upright type.

During the preparation of this paper, I examined the `toptesi` bundle, to find

```
\providecommand{\eu}{%
  \ensuremath{%
    {\mathop{\mathrm{e}}\nolimits}%
  }%
}
```

This is disputable under many respects:

- `\ensuremath` serves no real purpose;
- `\nolimits` can be safely omitted, because the `\mathop{...}` bit is followed by `}`, so surely there are no limits to take into account;
- `\mathop` itself is redundant, because the whole thing is braced, so it is treated as an ordinary symbol.

Oh, wait! No, `\mathop` is actually wrong! Consider the following code:

[6] The paper is also available in English (Guiggiani and Mori, 2008b).

```
\documentclass{standalone}
\usepackage{amsmath}
\newcommand{\euA}{\mathrm{e}}
\newcommand{\euB}{%
  \ensuremath{%
    {\mathop{\mathrm{e}}\nolimits}%
  }%
}
\begin{document}
$2\euA\euB$
\end{document}
```

The output is shown in figure 4. Do you see the problem? A single character in the argument to `\mathop` is raised or lowered so that it extends the same above and below the math axis.

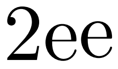


FIGURE 4: Magnified output for the Euler's constant problem

Now that we're on the spot, how to define a better `\tder` macro also supporting higher order derivatives? The first attempt,

```
\newcommand{\tder}[3][]{%
  \frac{\diff^{#1}#2}{\diff #3^{#1}}%
}
```

has a flaw: it unconditionally adds `\scriptspace` to both the numerator and denominator. If I measure the width of `\tder{f}{t}` in display math mode, with standard font and document class, I get `14.07712pt`; the version without the dummy exponents has width `11.91045pt`. More than two points! With the upright 'd', the difference would be half a point. And the visual result shows more:

$$\frac{d{}f}{d{}t} \neq \frac{df}{dt} \qquad \frac{\mathrm{d}{}f}{\mathrm{d}{}t} \neq \frac{\mathrm{d}f}{\mathrm{d}t}$$

Yes, we need to avoid the dummy superscript, also with the upright 'd', although the difference is less noticeable: we want perfect output, don't we? And we want macros that allow users to choose their own preferred 'd'. One could test whether the argument is empty, but there's a better way with xparse:

```
\NewDocumentCommand{\tder}{s o m m}{%
  \IfBooleanTF{#1}{\dfrac}{\frac}%
    {\diff\IfValueT{#2}{^{#2}}#3}% num
    {\diff #4\IfValueT{#2}{^{#2}}}% den
}
```

The *-version delivers `\dfrac` (just in case one needs it), otherwise `\frac` is used. The numerator and the denominator add the exponent only if the optional argument is specifically used. Thus `\tder{f}{t}` will not add a dummy exponent.

5 Sets, bras and kets

A short note to the title. Physicists have a sense of humor: a well-established notation for inner products is $\langle x \mid y\rangle$, called a "bracket". A mathematician would denote the linear or semilinear forms induced by the bracket as $\langle x \mid -\rangle$ and $\langle - \mid y\rangle$. Physicists, instead, use $\langle x|$ for the former and $|y\rangle$ for the latter, calling them "bra" and "ket".

Since several years, LATEX has been requiring e-TEX extensions, among which `\middle` is a very useful one. For instance, we can typeset

$$\left\{x \in \mathbf{R} \;\middle|\; -\frac{1}{2} \le x \le \frac{8}{5}\right\}$$

with no phantom and no null delimiter. On the other hand, the code

```
\left\{x\in\mathbf{R} \;\middle|\;
  -\frac{1}{2}\le x\le \frac{8}{5}\right\}
```

is still really ugly and something like

```
\set*{x\in\mathbf{R}\suchthat
  -\frac{1}{2}\le x\le \frac{8}{5}}
```

would be much nicer. We call xparse and expl3 to the rescue!

```
\documentclass[varwidth]{standalone}
\usepackage{amsmath}
\usepackage{xparse}

\ExplSyntaxOn
\NewDocumentCommand{\set}{som}
 {
  % limit the scope for \suchthat
  \group_begin:
  \cs_set_protected:Npn \suchthat
   {
    \tl_use:N \l__egreg_set_st_tl
   }
  \IfBooleanTF{#1}
   {
    \egreg_set_auto:n { #3 }
   }
   {
    \egreg_set_fixed:nn { #2 } { #3 }
   }
  \group_end:
 }

\tl_new:N \l__egreg_set_st_tl

\cs_new_protected:Nn \__egreg_set_st:n
 {
  \tl_set:Nn \l__egreg_set_st_tl { #1 }
 }

\cs_new_protected:Nn \egreg_set_auto:n
 {
  \__egreg_set_st:n
   {
    \nonscript\;
    \middle\vert
    \nonscript\;
   }
  \left\{ #1 \right\}
 }

\cs_new_protected:Nn \egreg_set_fixed:nn
 {
  \tl_if_novalue:nTF { #1 }
   {
    \__egreg_set_st:n { \mid }
    \lbrace #2 \rbrace
   }
   {
    \__egreg_set_st:n
     { \mathrel{#1\vert} }
    \mathopen{#1\lbrace}
    #2
    \mathclose{#1\rbrace}
   }
 }
\ExplSyntaxOff

\begin{document}

$\set{a,b,c}\cup\set[\big]{a,b,c}$

$\set{x\suchthat a<x<b}$

$\set[\Big]{x\suchthat a<x<b}$

$\set*{x\suchthat \dfrac{1}{2}<x<3}$

\end{document}
```

The idea is to use a syntax familiar from mathtools' `\DeclarePairedDelimiter`. The output is in figure 5.

$$\begin{gathered}\{a,b,c\}\cup\bigl\{a,b,c\bigr\}\\ \{x\mid a<x<b\}\\ \Bigl\{x\Bigm| a<x<b\Bigr\}\\ \left\{x\;\middle|\;\dfrac{1}{2}<x<3\right\}\end{gathered}$$

Figure 5: Examples of set notation

In the TeXbook, Knuth recommends to add thin spaces when the set builder notation contains a bar, that is, it is not just a list of elements. I disagree. How could it be implemented? It's possible to look for the presence of `\suchthat` at the outer level and, in this case, to add the thin spaces at either end; nested sets would examine their own contents for the presence at the outer level.

A full implementation would also feature the choice for the delimiter as a preamble setting. I leave this as an exercise for whoever wants to make a package out of this code.

There is some code duplication, but it's unavoidable. The reason is that using an `O{}` specifier for the optional argument would allow `\mathclose{#2\rbrace}` and no case distinction. However, one can see the difference if a subscript is added

`\rbrace_{1}` `\mathclose{\rbrace}_{1}`

$\}_1$ $\mathclose{\}}_1$

Different coding is possible, though. It would not be difficult to allow `|` instead of `\suchthat`. Look at how the macros for bras and kets can be defined.

```
\documentclass[varwidth{standalone}
\usepackage{amsmath}
\usepackage{xparse}

\NewDocumentCommand{\bra}{som}{%
  \IfBooleanTF{#1}
    {\left\langle #3 \right|}
    {%
     \IfNoValueTF{#2}
      {\langle#3\mathclose|}
      {\mathopen{#2\langle}#3\mathclose{#2|}}%
    }
}
\NewDocumentCommand{\ket}{som}{%
  \IfBooleanTF{#1}
    {\left\langle #3 \right|}
    {%
     \IfNoValueTF{#2}
      {\mathopen|#3\rangle}
      {\mathopen{#2|}#3\mathclose{#2\rangle}}%
    }
}

\NewDocumentCommand{\braket}{som}{%
  \IfBooleanTF{#1}
   {\extensiblebraket{#3}}
   {\fixedbraket{#2}{#3}}%
}

\ExplSyntaxOn
\NewDocumentCommand{\extensiblebraket}{m}
 {
  \group_begin:
  \char_set_active_eq:nN { ‘| } \egreg_bar_auto:
  \mathcode‘|="8000 \scan_stop:
  \left\langle
  #1
  \right\rangle
  \group_end:
 }

\NewDocumentCommand{\fixedbraket}{mm}
 {
  \group_begin:
  \char_set_active_eq:nN
    { ‘| }    % active char is |
    \egreg_bar_fixed: % equal to
  \mathcode‘|="8000 \scan_stop:
  \IfNoValueTF{#1}
   { \egreg_braket:n { #2 } }
   { \egreg_braket:nn { #1 } { #2 } }
  \group_end:
 }

\cs_new_protected:Nn \egreg_bar_auto:
 {
  \nonscript\,\middle\vert\nonscript\,
 }
\cs_new_protected:Nn \egreg_bar_fixed:
 {
  \mathinner{\egreg_size: \vert}
 }
\cs_new_protected:Nn \egreg_braket:n
 {
  \cs_set_protected:Nn \egreg_size: { }
  \langle #1 \rangle
 }
\cs_new_protected:Nn \egreg_braket:nn
 {
  \cs_set_protected:Nn \egreg_size: { #1 }
  \mathopen{\egreg_size: \langle}
  #2
  \mathclose{\egreg_size: \rangle}
 }
\ExplSyntaxOff

\begin{document}

$\bra{x}\quad\ket{x}$

$\braket{x|y}$

$\braket[\Big]{x|y|z}$

$\braket*{a|b}$

$\braket[\Big]{a|b|\dfrac{c}{d}}$

$\braket*{a|b|\dfrac{c}{d}}$

\end{document}
```

I'll not comment the code, except for mentioning how easy is to define the value of a character when it will be made active (math active, in this case).

$$\langle x|\quad|x\rangle$$
$$\langle x\,|\,y\rangle$$
$$\Bigl\langle x\Bigm|y\Bigm|z\Bigr\rangle$$
$$\langle a\,|\,b\rangle$$
$$\Bigl\langle a\Bigm|b\Bigm|\dfrac{c}{d}\Bigr\rangle$$
$$\left\langle a\middle|b\middle|\dfrac{c}{d}\right\rangle$$

Figure 6: Examples of bras and kets

6 Numbers and units

How should numbers be typed in the LaTeX document? Knuth himself once acknowledged that his usual practice is not very good and realized it when writing 'Concrete Mathematics' (Graham *et al.*, 1989), were numbers are typeset with the Euler font when they're used in their mathematical meaning (and not, say, as page markers).

When a number appears in text and is mentioned as a mathematical object is should be input inside a math formula:

```
a vector space of dimension~$5$
```

But what about large numbers that need to be split in smaller units for readability? For instance, can you spell out 7400043022221 without first counting how many digits the number has? Isn't 7 400 043 022 221 easier to parse? Possibly not for an American who's more accustomed to 7,400,043,022,221 (and probably would be at stake when people talks about meters and liters).

Now let's face a problem: your scientific paper has several tables with numeric data and you're not sure about the editorial policy of the journal you'll be submitting it. Will the journal require American style or prefer thin spaces for grouping digits?

Table 1: Tables with different formatting options for numbers (Source: Mr Leporello, private communication)

Nation	Number	Nation	Number
Italy	640 375	Italy	640,375
Germany	231 803	Germany	231,803
France	100 002	France	100,002
Turkey	91 329	Turkey	91,329
Spain	1 003 000	Spain	1,003,000

Let's consider the two tables in table 1. They are typeset with exactly the same input, namely

```
\begin{tabular}{
  @{}
  l
  S[table-format=7.0]
  @{}
}
\toprule
Nation   & {Number} \\
\midrule
Italy    &  640375 \\
Germany  &  231803 \\
France   &  100002 \\
Turkey   &   91329 \\
Spain    & 1003000 \\
\bottomrule
\end{tabular}
```

and it's siunitx (Wright, 2018) doing all the magic. Of course there is a catch: just before the second copy of the table I added

```
\sisetup{group-separator={,}}
```

I could have added the option also in the bracketed argument to the `S` column, which is one of the facilities made available by the package. Similarly, the big number above has been typeset first with `\num{7400043022221}` and then with

```
\num[group-separator={,}]{7400043022221}
```

The default for the package is to use a thin space as a group separator between digits. An `S` column basically applies `\num` to every entry, but also aligns them at the decimal separator. In the case of our Leporello table, all entries are integer, so they're right aligned.

If an entry belonging to an `S` column is braced, it will be ignored as far as number alignment is concerned and centered on the total width of the column (options are available for left or right alignment). This is obviously needed in the header.

With another option we can easily scale down the figures:

Nation	Number
Italy	640×10^{3}
Germany	232×10^{3}
France	100×10^{3}
Turkey	91.3×10^{3}
Spain	1.00×10^{6}

This is achieved with the options

```
\sisetup{
  round-mode=figures,
  round-precision=3,
  scientific-notation=engineering
}
```

and by changing the column specifier to

```
S[table-format=3.2e1]
```

which directs to reserve space for three digits in the integer part, two in the mantissa and one in the exponent. The table body in the input has not changed in any way.

One might write an entire large chapter of the LATEX Companion about siunitx. Some time ago, Joseph Wright took up the job of making a successor package to SIunit adding some features along the way. For version 2 he had the idea of exploiting expl3 which not only allowed for *many* more features and facilities, but made him enter the LATEX team.[7] He's into chemistry and tables with numeric data are his staple food.

The main purpose of the package is of course typesetting numbers with their SI unit according to the guidelines of the Bureau International des Poids et Mesures (BIPM). This is also part of the ISO standard mentioned before:

```
\SI{1}{\newton} is defined as
\SI{1}{\kilogram\meter\per\second\squared}
```

will typeset "1 N is defined as 1 kg m s^{-2}". However, if we prefer slashes instead of negative exponent, we can add to the preamble

```
\sisetup{per-mode=symbol}
```

and the same text will now typeset as "1 N is defined as 1 kg m/s^{2}". The mode can also be changed on a local basis with an optional argument to `\SI`.

All SI units and prefixes are supported:

```
\SI{5}{\tera\meter} \SI{2}{\pico\farad}
```

yields 5 Tm and 2 pF. One can also print just a unit with `\si`: the unit for energy is the J which is the same as kg m^{2} s^{-2}.

Going on with our fictional scientist who's uncertain where her breakthrough paper will be published, decimal numbers might require the period as separator, or the comma; if in scientific notation there could be the $\times 10^{n}$ part or En might be asked for. How to do it? Not to mention uncertainty! Let's take as an example the rest mass of the electron

$$9.109\,383\,701\,5(28) \times 10^{-31}\,\mathrm{kg}$$

$$9.1093837015(28) \times 10^{-31}\,\mathrm{kg}$$

$$9{,}109\,383\,701\,5(28) \times 10^{-31}\,\mathrm{kg}$$

$$(9.109\,383\,701\,5 \pm 0.000\,000\,002\,8) \times 10^{-31}\,\mathrm{kg}$$

$$9.109\,383\,701\,5 \times 10^{-31}\,\mathrm{kg}$$

$$9.109\,383\,701\,5(28)\mathrm{E}{-31}\,\mathrm{kg}$$

The first line has been input with

```
\SI{9.1093837015(28)E-31}{\kilogram}
```

and the following lines by adding an option

```
\SI[⟨option⟩]{9.1093837015(28)E-31}
  {\kilogram}
```

The used options are, in order,

```
output-decimal-marker={,}
group-digits=integer
separate-uncertainty
omit-uncertainty
output-exponent-marker=\mathrm{E}
```

and they can be combined to get the desired effect *without changing the code in the document* if the settings are done with `\sisetup` in the document preamble. When inputting numbers, one can use spaces and either a decimal period or decimal comma. The first mandatory argument to `\SI` behaves the same as the mandatory argument to `\num`, so I'll use the latter:

```
\num{12345.678}
\num{12345,678}
\num{12 345.678}
```

will print the same

$$12\,345.678 \quad 12\,345.678 \quad 12\,345.678$$

Very few things are hardwired in siunitx: one can instruct it to ignore something, for instance. Suppose you have a set of numbers with comma separators

[7] It seems that understanding and propagating expl3 opens a straight way to the team.

for groups: the big number already used might be available as `7,400,043,022,221`. Set (globally or locally) the `input-ignore` option and remove the comma from the possible decimal separators:

```
\num[
  input-ignore={,},
  input-decimal-markers={.}
]{7,400,043,022,221}
```

and you’ll get

$$7\,400\,043\,022\,221$$

The package is not limited to ‘standard numbers’: it also copes with angles, time and complex numbers. For instance, we can type

```
\ang{30.24}
\ang{30;12;44.375}
\num{3-4i}
```

to get

$$30.24^\circ,\quad 30^\circ 12' 44.375'',\quad 3-4\mathrm{i}$$

Oh, dear! An upright ‘i’! Let’s fix it with

```
\sisetup{
  output-complex-root=\mathnormal{i}
}
```

(one could also tell it to use ‘j‘, of course) and get

$$3-4i$$

Phew! Yes, the package obviously adheres to the ISO standard, but it’s very customizable.

7 Further reading

Twenty-two years have passed from the seminal paper by Claudio Beccari: we have seen great progress in the field of math typesetting, in particular towards the uniformity that’s necessary in technical and commercial reports.

There are other packages that can be tried for the purpose of compliance to the ISO 80000-2:2009 standard. I would mention isomath by Günter Milde (Milde, 2012) and also unicode-math by Will Robertson (Robertson, 2019) that provided facilities to the purpose; the former is for legacy `pdflatex`, the latter for XƎLATEX and LuaLATEX.

References

Beccari, Claudio (1997). «Typesetting mathematics for science and technology according to ISO 31/XI». *TUGboat*, **18** (1).

Dieudonné, Jean (1972). *Treatise on Analysis. Vol. III.* Academic Press, New York-London. Translated from the French by I. G. MacDonald, Pure and Applied Mathematics, Vol. 10-III.

Graham, Ronald L., Donald E. Knuth and Oren Patashnik (1989). *Concrete mathematics.* Addison-Wesley Publishing Company, Advanced Book Program, Reading, MA. A foundation for computer science.

Gregorio, Enrico (2009). «Simboli matematici in TEX e LATEX». *ArsTEXnica*, **8**, pp. 7–24. `http://www.guitex.org/home/numero-8`.

Guiggiani, Massimo and Lapo F. Mori (2008a). «Consigli su come *non* maltrattare le formule matematiche». *ArsTEXnica*, **5**, pp. 5–14. `http://www.guitex.org/home/numero-5`.

— (2008b). «Suggestions on how *not* to mishandle mathematical formulae». *TUGboat*, **29** (2).

Milde, Günter (2012). «isomath — mathematical style for science and technology». `https://ctan.org/pkg/isomath`. Version 0.6.1, `texdoc isomath`.

Robertson, Will (2019). «Experimental Unicode mathematical typesetting: The unicode-math package». `https://ctan.org/pkg/unicode-math`. Version 0.8o, `texdoc unicode-math`.

Rudin, Walter (1966). *Real and complex analysis.* McGraw-Hill Book Co., New York-Toronto, Ont.-London.

— (1974). *Analisi Reale e Complessa.* Bollati Boringhieri.

Tellechea, Christian (2019). «L’extension pour TEX et LATEX systeme». `https://ctan.org/pkg/systeme`. Version 0.32, `texdoc systeme`.

Wright, Joseph (2018). «siunitx — A comprehensive (SI) units package». `https://ctan.org/pkg/siunitx`. Version 2.7s, `texdoc siunitx`.

▷ Enrico Gregorio
Dipartimento di Informatica, Università di Verona
`enrico dot gregorio at univr dot it`

Bibliographies, LaTeX and friends

Guido Milanese

Abstract

This lecture deals with the treatment of bibliographies within a LaTeX framework and workflow. A comparison of BibTeX bibliography format with other widely used formats shows that BibTeX has several advantages. The classical bibtex programme is now obsolete, and biblatex + biber offer a highly customisable choice for bibliographies in any research area. GUI environments are also discussed, as well as possible future developments.

Sommario

Questa conversazione si occupa del trattamento delle bibliografie all'interno di un ambiente di lavoro basato su LaTeX. Una comparazione del formato bibliografico di BibTeX con altri formati usati ampiamente mostra che BibTeX offre parecchi vantaggi comparativi. Il classico programma bibtex è da ritenersi obsoleto, poiché biblatex + biber offrono un ambiente altamente flessibile per generare bibliografie in ogni area di ricerca. Vengono anche discussi agli ambienti grafici e le prospettive per il futuro.

1 BibTeX and other formats

Let us consider a working example. One of the most important online catalogues is "Library Hub Discover", a British online resource that has recently replaced Copac and SUNCAT, offering access to 117 catalogues for more than 40 million of bibliographic items (`https://discover.libraryhub.jisc.ac.uk`). "Library Hub Discover" (henceforth referred to as "LHB") gives the user the opportunity of exporting bibliographic items into several widely used formats. Let us compare the BibTeX file generated by LHB with the Endnote–Zotero file generated by the same online resource (actually it is a RIS card[1]: see fig. 1. At first glance, the two formats show the same amount of information (the abstract field is missing from RIS, but this is a problem of the filter used by "Library Hub Discover"); BibTeX, however, always generates a "label" for any item – this is not an option; this feature is very important in order to link items, which makes possible to biblatex to generate highly sophisticated bibliographic entries. The three formats referred to (RIS, Endnote/Zotero, and BibTeX) are widely used on the Internet, and it is fairly easy to convert among them: BibTeX is known e.g. to Google Books and Google Scholar. From a practical point of view, the three formats are equally successful.

2 The BibTeX format

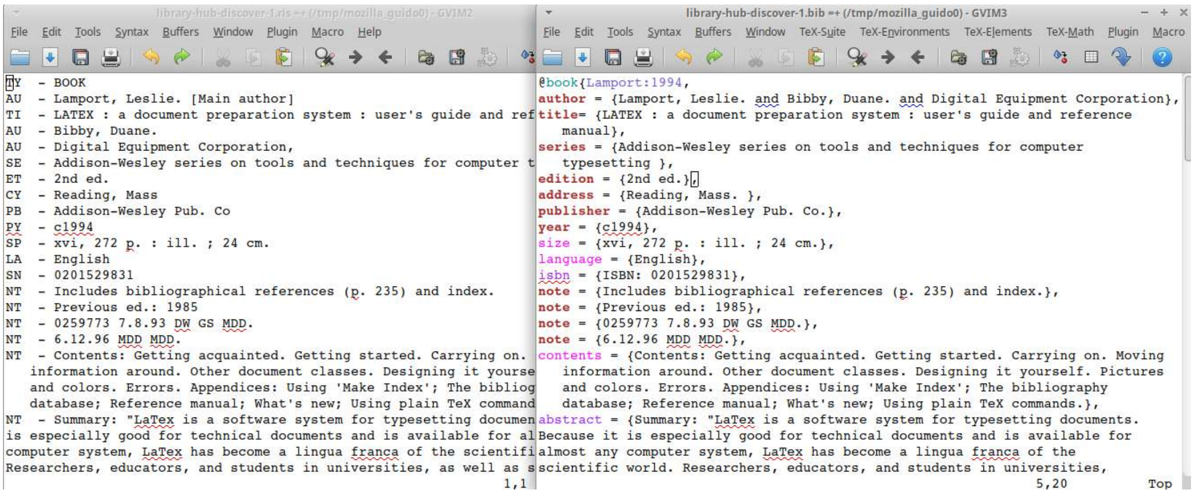
As an example of BibTeX, we can use a very basic card generated by LHB:

```
 1 @book{Lamport:1994,
 2 author = {Lamport, Leslie.},
 3 title= {LATEX : a document preparation system},
 4 edition = {2nd ed.},
 5 address = {Reading, Mass. },
 6 publisher = {Addison-Wesley},
 7 year = {1994},
 8 language = {English},
 9 isbn = {ISBN: 0201529831},
10 isbn = {ISBN: 9780201529838},
11 note = {User's guide and reference manual.},
12 location = {University of Sussex Library},
13 }
```

There are some minor mistakes (see section 9); now, let us read the information provided. Line 1 is very important, providing a unique label for each entry; think of this line as the number plate of a car: you can have an archive of thousands of entries, but each one must have a unique label. Lines 2 to 10 feature the most important bibliographic information, such as author, title, and so on; lines 9 and 10 are not a useless duplicate, because the two ISBN code formats are listed, the former of 10 digits, and the latter of 13 digits. It is worth mentioning that some of the fields listed in this card are actually biblatex fields, unknown to the original BibTeX format: I refer to the "ISBN", "language" and "location" fields. Line number 11 is used by the LHB filter as a sort of black hole, because in this case the "note" field is used for what is really the subtitle of Lamport's book. The last line informs of the library location of this book. All the fields must be balanced: each piece of information must be enclosed in curly braces and followed by a comma, and the whole entry must be opened and closed by a curly braces pair. Uppercase and lowercase are immaterial: `AUTHOR` and `author` and `Author` are valid and can be mixed freely. Consistency is recommended for stylistic reasons, but it does not change the behaviour of programmes using this format.

There are other fields that can prove very useful to scholars, such as `abstract`, `keywords`, `annotation`… Any parser using BibTeX files can perform powerful researchers, combining the infor-

[1] See `https://en.wikipedia.org/wiki/RIS_(file_format)`; `https://en.wikipedia.org/wiki/EndNote`; `https://en.wikipedia.org/wiki/Zotero`.


library-hub-discover-1.ris =+ (/tmp/mozilla_guido0) - GVIM2
File Edit Tools Syntax Buffers Window Plugin Macro Help
library-hub-discover-1.bib =+ (/tmp/mozilla_guido0) - GVIM3
File Edit Tools Syntax Buffers Window TeX-Suite TeX-Environments TeX-Elements TeX-Math Plugin Macro

```
TY  - BOOK
AU  - Lamport, Leslie. [Main author]
TI  - LATEX : a document preparation system : user's guide and ref
AU  - Bibby, Duane.
AU  - Digital Equipment Corporation,
SE  - Addison-Wesley series on tools and techniques for computer t
ET  - 2nd ed.
CY  - Reading, Mass
PB  - Addison-Wesley Pub. Co
PY  - c1994
SP  - xvi, 272 p. : ill. ; 24 cm.
LA  - English
SN  - 0201529831
NT  - Includes bibliographical references (p. 235) and index.
NT  - Previous ed.: 1985
NT  - 0259773 7.8.93 DW GS MDD.
NT  - 6.12.96 MDD MDD.
NT  - Contents: Getting acquainted. Getting started. Carrying on.
   information around. Other document classes. Designing it yourse
   and colors. Errors. Appendices: Using 'Make Index'; The bibliog
   database; Reference manual; What's new; Using plain TeX command
NT  - Summary: "LaTex is a software system for typesetting documen
is especially good for technical documents and is available for al
computer system, LaTex has become a lingua franca of the scientifi
Researchers, educators, and students in universities, as well as s
                                                              1,1
```

```
@book{Lamport:1994,
author = {Lamport, Leslie. and Bibby, Duane. and Digital Equipment Corporation},
title= {LATEX : a document preparation system : user's guide and reference
   manual},
series = {Addison-Wesley series on tools and techniques for computer
   typesetting },
edition = {2nd ed.},
address = {Reading, Mass. },
publisher = {Addison-Wesley Pub. Co.},
year = {c1994},
size = {xvi, 272 p. : ill. ; 24 cm.},
language = {English},
isbn = {ISBN: 0201529831},
note = {Includes bibliographical references (p. 235) and index.},
note = {Previous ed.: 1985},
note = {0259773 7.8.93 DW GS MDD.},
note = {6.12.96 MDD MDD.},
contents = {Contents: Getting acquainted. Getting started. Carrying on. Moving
   information around. Other document classes. Designing it yourself. Pictures
   and colors. Errors. Appendices: Using 'Make Index'; The bibliography
   database; Reference manual; What's new; Using plain TeX commands.},
abstract = {Summary: "LaTex is a software system for typesetting documents.
Because it is especially good for technical documents and is available for
almost any computer system, LaTex has become a lingua franca of the
scientific world. Researchers, educators, and students in universities,
                                                        5,20          Top
```

FIGURE 1: Endnote/Zotero and BIBTEX formats

mation provided by fields – such as a query looking for of books concerning typesetting (`keywords` field) published between 2010 and 2018 (`year` field), and featuring the word «useless» in the `annotation` field. This research will extract all the books about typesetting, published between 2010 and 2018, regarded as useless by the owner of the database. There is of course great freedom in the use of "personal" fields.

3 Bibliographic data and output formats

BIBTEX fields are in fact a particular format of tags. It is indeed quite simple to export a BIBTEX file into another format called BIBTEXML – a format designed in 2007 but not very successful[2]:

```
<bibtex:entry xmlns:bibtex="http://bibtexml.sf.net/"
   bibtex:id="Lamport:1994">
<bibtex:book>
<bibtex:author>Lamport, Leslie. </bibtex:author>
<bibtex:title>LATEX : a document preparation system
   </bibtex:title>
<bibtex:edition>2nd ed.</bibtex:edition>
<bibtex:address>Reading, Mass. </bibtex:address>
<bibtex:publisher>Addison-Wesley </bibtex:publisher>
<bibtex:year>1994</bibtex:year>
<bibtex:language>English</bibtex:language>
<bibtex:isbn>ISBN: 0201529831</bibtex:isbn>
<bibtex:isbn>ISBN: 9780201529838</bibtex:isbn>
<bibtex:note>User’s guide and reference manual.
   </bibtex:note>
<bibtex:location>University of Sussex Library
   </bibtex:location>
</bibtex:book>
</bibtex:entry>
```

The information provided by this XML format is exactly the same featured by the internal tagging system of BIBTEX – just heavier and more difficult to read; BIBTEX is simply a tagging system, just a very light and simple one. The purpose of any tagging system, as XML which is familiar to anyone using a computer nowadays, is to separate data from representation: BIBTEX is no exception: there are literally thousands of bibliographic formats required by publishers and journals, and it is vital to keep information on a book, article or whatever separated from the output needed for a particular publication. In this incredible variety of output formats, a very basic arrangement can be the following one:

1. the number reference system: each item referred to in a book or article is labelled with a number. At the end of the text, a list will show the complete bibliographic information. For example, a citation from page 32 of Lamport's book, provided it receives e.g. number 6 in the final list, will be «[6, 32]».

2. the author-year systems: from the BIBTEX file, fields `author` and `year` will be extracted and citations will take this form: «Lamport 1994, 32» or «Lamport, 1994, 32» or «LAMPORT 1994, 32»... Again, at the end of the book or article the reader finds the complete information about the titles referred to. If there are more than one title published by the same author in the same year, the year field is completed with a letter, such as, e.g., «Lamport 1994a, 32».

3. the author-title systems: from the BIBTEX file, fields `author` and `title` will be extracted and citations will take this form: «Lamport, *LATEX*, 32», or a variety of this scheme. For this kind of system, it is advisable to add a `shorttitle` field, in order to obtain reasonably short references;

4. the varieties of the "verbose" systems, very popular in the humanities, particularly in Italy. Generally, the first citations is complete, given

[2] See `https://sourceforge.net/projects/bibtexml/files/BibTeXConverter/`.

in a footnote, and the following ones take the form of an author-title system. Normally with this kind of reference system a final bibliography is not required by journals, although it remains necessary for books.

4 BibTeX, bibtex, biber and biblatex

Since 1985, at the very beginning of LaTeX, the programme bibtex (with the same name of the bibliographic format) was designed by Lamport himself and Oren Patashnik in order to obtain from the output format agnostic bibliographic files provided by BibTeX the kind of output format required by the final user, i.e. by the journal or publishing house[3]. This programme never reached version 1.0 – the most recent version at CTAN (July 2019) is version 0.99d. bibtex is a good piece of software, but at the beginning of this century it was clear that it had serious troubles with Unicode and that it was not easily customisable. There were several attempts to improve the situation[4]: a very promising tool is (was) bibulous, a Python script with a simple, intuitive approach to develop new styles – but apparently the development ceased years ago[5]. The standard replacement for bibtex is now biber + biblatex[6]. The former is used to sort data and to deal with labels, while all the process of generating the desired output format is performed by biblatex. Advantages of the new pair of programmes in comparison with old bibtex are, from the point of view of the final user:

- no problems with Unicode – sorting can be adapted to the language of the document and also single entries can be made aware e.g. of the hyphenation required by the language of that particular entry;
- it is possible to build new styles, which is particularly important in countries, such as Italy, where each publisher wants to receive files formatted according to the particular styles of this of that particular journal or series;
- more flexible use of the `crossref` field;
- full support for related entries, e.g. "reprinted as. . . " or "translation of. . . ";
- the new `XDATA` field, a sort of container for one or more fields, e.g. name of a publishing house and place of publication, or name of a series of books, publishing house, place of publication. This can be convenient to avoid useless repetition of information in a file.
- suppression/inclusion of fields is straightforward, even for some entry types.

No heavy tailoring of old BibTeX files is necessary to use the new features. However, to take full advantage from biber + biblatex some additional fields are advisable, and some old field names must be adapted. If we take the basic example downloaded from LHB (see sec. 1) we could adapt it as follows:

```
@book{Lamport:1994,
author = {Lamport, Leslie.},
title= {{LATEX} A document preparation system},
EDITION = {2},
LOCATION = {Reading, Mass. },
publisher = {Addison-Wesley},
year = {1994},
LANGID = {AMERICAN},
language = {english},
isbn = {ISBN: 0201529831},
isbn = {ISBN: 9780201529838},
subtitle = {User's guide and reference manual},
LIBRARY = {University of Sussex Library},
}
```

I have capitalised the fields that have to be adapted to use with biber + biblatex. The `edition` field must contain only a number, in order to format the field according to the needs of languages and styles, e.g.: «1994, 2nd ed.», or «II edizione, 1994», or «1994^2»; the new field `langid` is used by biblatex in order to format entries according to hyphenation and, if required, to all the rules of the language of that particular entry: for example the strings used to replace "translation" or "reprint" in languages other than American English. The old field `language` can still be used for the language of the work, but will not affect the rules for hyphenation etc. Since British English and American English have their local rules, it is advisable to use «american» or «british» for the `langid` field, avoiding the generic label «english». One possible ambiguity is the field `location:` traditional BibTeX used it for library location, while biblatex uses the same field name as an alias for `address`; the name of libraries holding the book is archived as `library`. In this example, the field `titleaddon` is used to describe the real hierarchy of Lamport's book title.

5 How to survive BibTeX

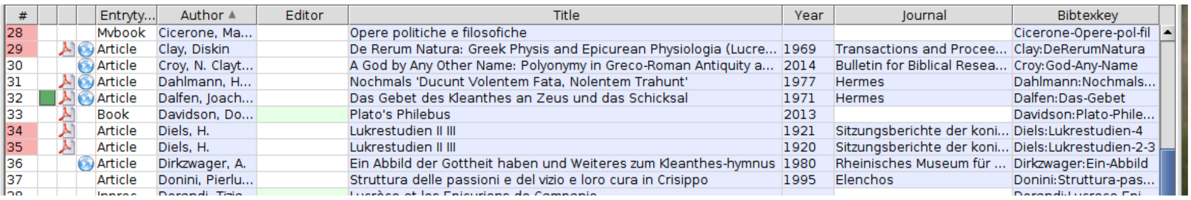
As it happens with any tagging system, BibTeX files are error-prone. Even if good text editors, such as (g)vim, do an excellent job checking syntax, it is too easy to forget to close a bracket or to add a comma at the end of a field. A good GUI interface makes for BibTeX makes the users' life much easier: see `https://en.wikipedia.org/wiki/Comparison_of_reference_management_software` for a comparison of the features offered by reference management programmes. Many among them have a Free Software licence, and the great majority of these programmes either use

[3] See Lamport (1994, 69-72) and more recently Mittelbach *et al.* (2004, 683-812).

[4] See Hufflen (2011).

[5] See `https://pypi.org/project/bibulous`. The latest version is dated 2015.

[6] A recent updated manual is Voss (2011); on biblatex, see Lehman's manual (Lehman *et al.*, 2018).



#				Entryty...	Author ▲	Editor	Title	Year	Journal	Bibtexkey
28				Mvbook	Cicerone, Ma...		Opere politiche e filosofiche			Cicerone-Opere-pol-fil
29		📄	🌐	Article	Clay, Diskin		De Rerum Natura: Greek Physis and Epicurean Physiologia (Lucre...	1969	Transactions and Procee...	Clay:DeRerumNatura
30			🌐	Article	Croy, N. Clayt...		A God by Any Other Name: Polyonymy in Greco-Roman Antiquity a...	2014	Bulletin for Biblical Resea...	Croy:God-Any-Name
31		📄	🌐	Article	Dahlmann, H...		Nochmals 'Ducunt Volentem Fata, Nolentem Trahunt'	1977	Hermes	Dahlmann:Nochmals...
32	🟩	📄	🌐	Article	Dalfen, Joach...		Das Gebet des Kleanthes an Zeus und das Schicksal	1971	Hermes	Dalfen:Das-Gebet
33		📄		Book	Davidson, Do...		Plato's Philebus	2013		Davidson:Plato-Phile...
34		📄		Article	Diels, H.		Lukrestudien II III	1921	Sitzungsberichte der koni...	Diels:Lukrestudien-4
35		📄		Article	Diels, H.		Lukrestudien II III	1920	Sitzungsberichte der koni...	Diels:Lukrestudien-2-3
36			🌐	Article	Dirkzwager, A.		Ein Abbild der Gottheit haben und Weiteres zum Kleanthes-hymnus	1980	Rheinisches Museum für ...	Dirkzwager:Ein-Abbild
37				Article	Donini, Pierlu...		Struttura delle passioni e del vizio e loro cura in Crisippo	1995	Elenchos	Donini:Struttura-pas...
38				Inproc...	Dorandi, Tizia...		Lucrèce et les Epicuriens de Campanie			Dorandi:Lucrece-Epi...

Figure 2: Jabref graphical user interface

BibTeX files or can import/export from this format – a choice is basically a matter of preferences and of personal taste. However, I think that some points may be taken into serious account:

- some very popular programmes are proprietary software. Those who prefer to use Free Software would probably avoid using for example Mendeley (https://www.mendeley.com);
- some programmes offer the user an online space to archive data. Those who prefer local storage should consider this point; see again Mendeley;
- does a programme use BibTeX to archive data, or just to import/export data? For example, Zotero can archive data on local storage, but using its own SQL format. Programmes using BibTeX directly have the great advantage of archiving data in a plain text file, that is not bound to a particular GUI programme and to its format;
- those users who work on several platforms, e.g. Windows at home and Linux at work, will probably prefer programmes being actively developed for different platforms. For example, pybibliographer (https://pybliographer.org) runs only under Linux, and this is an obvious problem for many users.

At the moment of writing, Jabref is probably the most complete GUI environment for using BibTeX files (http://www.jabref.org/). It offers a good support also for the new features provided by biblatex and links very effectively BibTeX entries to local or remote files, such as PDF files or other forms of archived information, and to texts written by users as their own comments. See fig. 2: the green square shows that a file containing notes by the user is available; the traditional PDF icon opens a local PDF; the third icon, with the form of a little globe, opens a link on the web. Even if these features are offered by other programmes, the advantages of Jabref are

1. to use a simple BibTeX file to archive information
2. to be platform-independent.

In other words, while a Zotero archive needs Zotero to work properly, the work done by Jabref can be edited and modified using any other GUI interface, such as e.g. pybibliographer, or any text editor, even Notepad; the file produced is a completely transparent, standard BibTeX archive.

Citing an entry from a BibTeX archive is made easy by all these GUI interfaces, either clicking on a particular item or hitting a combination of keys. To cite page 32 of Lamport's book, the LaTeX file will feature the required citation in the neutral form `\cite[32]{Lamport:1994}`. In the final output, the citation will be formatted, as previously described, according to the desired style.

6 Using BibTeX to build a scholarly archive

Since BibTeX entries are highly customizable, it is possible to use this format to build a complete environment to keep trace of one's scholarly work and make it available in the future. Let me use a real example taken from one of my databases:

```
@ARTICLE{Dalfen:Das-Gebet,
  author = {Dalfen, Joachim},
  journal = {Hermes},
  keywords = {fant, fhel, stoa},
  pages = {174-183},
  shorttitle = {{D}as {G}ebet des {K}leanthes},
  title = {{D}as {G}ebet des {K}leanthes an
      {Z}eus und das {S}chicksal},
  volume = {99},
  langid = {german},
  year = {1971},
  annotation = {2018-Cleante},
  file = {Dalfen\:Das-Gebet.md:Neworbis/
    Dalfen\:Das-Gebet.md:Markdown},
  issn = {00180777},
  pdf = {Dalfen:Das-Gebet.pdf},
  timestamp = {24.03.2018},
  url = {http://0-www.jstor.org.opac.
  unicatt.it/stable/4475677}
}
```

I am using the `annotation` field to archive a keyword showing for which publication I have used this particular article or book; I keep a list of these keywords in a separate file and in the future I will be able to know which publications I had been using for this particular research. The field `timestamp` is useful to know when I read and archived this work; the field `keywords` is obviously useful if and only if the user is consistent in naming keywords: if, for example, I would sometimes use "stoa" and sometimes "stoicism", an attempt to extract all the entries concerning Stoicism would fail. This is one of the instances where consistency is crucial for a successful use of this format.

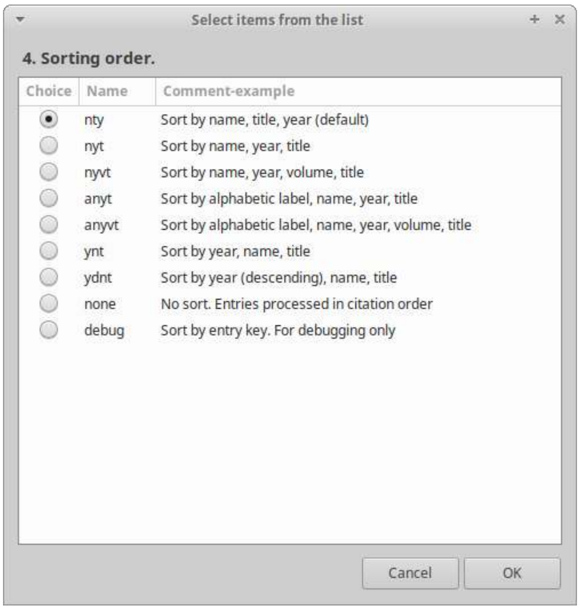
«Digital tools have yet to develop models for displaying and replicating the self-reflexive operations of bibliographical tools, which alone are operations for thinking and communicating – which is to say, for transforming storage into memory, and data into knowledge. We have to design and build digital environments for those purposes» (McGann, 2016, 363). I think that an imaginative use of BibTEX archives already offers a serious answer to these questions.

7 The **philosophy** package

A tremendous amount of bibliographic styles has been produced by journals and publishers: more than 9000 "Citation Style Language" (CSL) bibliographic styles are now available (`https://citationstyles.org/`). The CSL format is used by bibliography managers such as Zotero and Mendeley: unfortunately, BibTEX users are offered a good choice of styles, but not comparable to this incredible wealth. The Italian scholar Ivan Valbusa has developed a package aimed particularly at the needs of humanists, who every day struggle with the various and exasperating bibliographic requirements of publishers and journals. This package, called biblatex-philosophy (`https://ctan.org/pkg/biblatex-philosophy`), now at version 1.9.8a, features three basic styles, with a rich offer of options, whose combination and selection is likely to face, if not all, at least the large majority of the requirements of journals and publishers[7].

8 The **zblbuild** package

This utility offers a GUI aimed at helping the generic user building his own bibliographic format. A series of GUI widgets (question and checkboxes) select among the various possible formats and generate a biblatex call featuring all the necessary options. The programme is particularly tailored to work along with Valbusa's philosophy package[8]. See an example at fig. 3.



Select items from the list

4. Sorting order.

Choice	Name	Comment-example
●	nty	Sort by name, title, year (default)
○	nyt	Sort by name, year, title
○	nyvt	Sort by name, year, volume, title
○	anyt	Sort by alphabetic label, name, year, title
○	anyvt	Sort by alphabetic label, name, year, volume, title
○	ynt	Sort by year, name, title
○	ydnt	Sort by year (descending), name, title
○	none	No sort. Entries processed in citation order
○	debug	Sort by entry key. For debugging only

Figure 3: Blbuild – a widget

9 Other utilities

BibTEX is a complex system, and in more than 30 years of life, which is a geological era in this field, it has generated many utilities to maintain archives or to adapt them to new formats such as biblatex. Although a couple of dozens of these utilities are listed by CTAN, it is worth noticing that many among them are old, written even more than 25 years ago, and cannot be useful within a modern framework. The most complete I know among these utilities is called BibTool, developed and maintained by Gerd Neugebauer since 1995. This rich programme features a wide variety of utilities: sort an archive, extract items referred to in an article or book, rename fields – for example, the `location` field of legacy BibTEX files that is to be renamed to `library` if the archive is going to be used by biblatex. A real "must": the list of possible manipulations listed in the CTAN page are the following ones[9]:

- Pretty-printing data bases;
- Syntactic checks with error recovery;
- Semantic checks;
- Sorting and merging of data bases;
- Generation of uniform reference keys according to predefined rules or according to user specification;
- Selecting references used in one publication which are found by analysing an "aux" file;
- Controlled rewriting of fields utilising regular expressions to specify the rewriting rules;
- Macro (String) expansion to eliminate the need of extra string definitions;
- Collecting statistics about one or more databases.

[7] See Valbusa's manual on CTAN: Valbusa (2018), and – for a previous version – Valbusa (2010).

[8] See `https://ctan.org/pkg/zblbuild`. For a previous version see Milanese (2015).

[9] See `https://www.ctan.org/pkg/bibtool`.

There are also many other little utilities that may prove worth trying. For example, frequently, BibTeX entries produced by automatic systems such as Google or Library Hub Discover contain little mistakes (spurious spaces, wrong labels) that should be manually corrected. The programme copac-clean (labelled from the old name of Library Hub Discover) improves the output produced by those systems, cleaning minor errors and adapting field names such as `location` and `language` to the needs of a present-day biblatex installation[10].

10 Using BibTeX without LaTeX

BibTeX was designed to be used along with LaTeX, but it can be effectively employed also within other environments and workflows because of its very simple tagged structure. The best example I know is offered by markdown, particularly in its pandoc variety. This format is very easy to read, needs no particular piece of software to edit files, and *via* the pandoc filter can generate a variety of output formats, including HTML, XML-TEI, ODT, DOCX, and LaTeX. Direct generation of a PDF file is possible if a TeX system is installed[11].

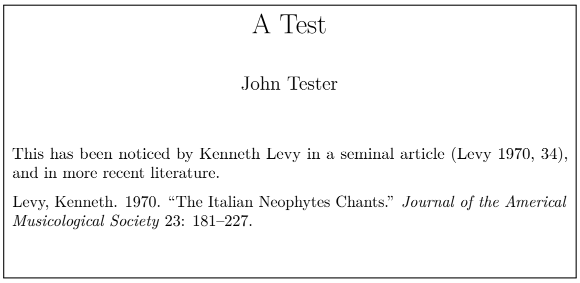
The interesting point is that pandoc can make use of a standard BibTeX archive but compile a PDF file using a CSL style format (see section n. 7). If preferred, the user can compile the bibliography calling biber + biblatex or even legacy bibtex (which is clearly not advisable). For example, consider this minimal MD file:

```
---
title: A Test
author: John Tester
bibliography: 'test.bib'
---

This has been noticed by Kenneth Levy in a
seminal article [@Levy:ItalianNeophytes 34],
and in more recent literature.
```

Running pandoc as follows, the filter defaults to a generic author–year style (backslash for readability only):

```
pandoc --filter pandoc-citeproc\
bibliotest.md  -s -o bibliotest.pdf
```

The final output is displayed at fig. 4. The MD file is processed by pandoc, silently calling one of the varieties of LaTeX compilers. Since no option is listed, pdflatex and the standard fonts are used, but the bibliography is produced calling the pandoc-citeproc filter, not biber + biblatex. Adding the name of a particular CSL style its features are used by the bibliographic filter, e.g.:



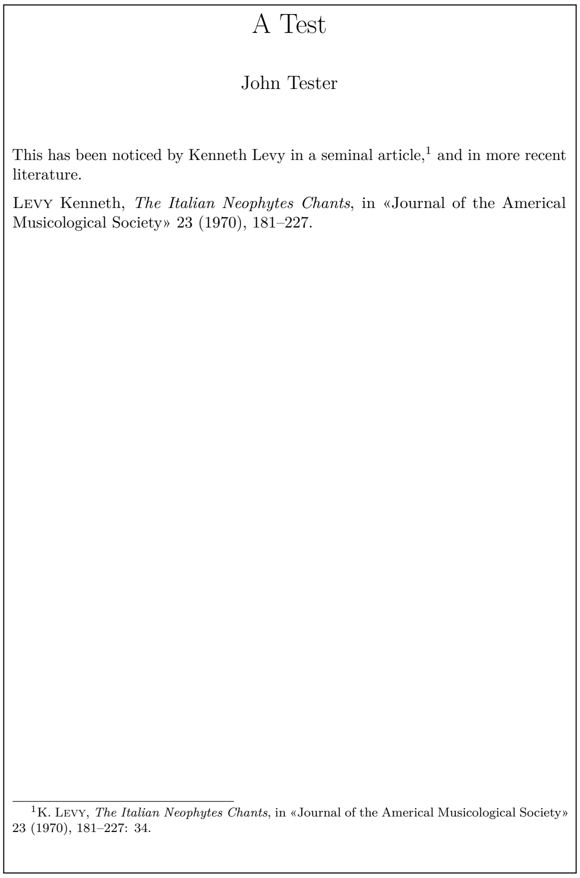
A Test

John Tester

This has been noticed by Kenneth Levy in a seminal article (Levy 1970, 34), and in more recent literature.

Levy, Kenneth. 1970. "The Italian Neophytes Chants." *Journal of the Americal Musicological Society* 23: 181–227.

Figure 4: Pandoc + LaTeX: default CSL



A Test

John Tester

This has been noticed by Kenneth Levy in a seminal article,[1] and in more recent literature.

Levy Kenneth, *The Italian Neophytes Chants*, in «Journal of the Americal Musicological Society» 23 (1970), 181–227.

[1] K. Levy, *The Italian Neophytes Chants*, in «Journal of the Americal Musicological Society» 23 (1970), 181–227: 34.

Figure 5: Pandoc + LaTeX: verbose CSL

```
pandoc --filter pandoc-citeproc\
bibliotest.md --csl=universita-pontificia\
-salesiana-it.csl -s -o bibliotest.pdf
```

This is an Italan "verbose" style, and the filter instructs LaTeX accordingly: see fig. 5. Pandoc can also use biber + biblatex: which one is the better? The pandoc-citeproc filter is attractive because it can make use of the huge collection of CSL styles; on the other hand, biber + biblatex offer a level of fine-tuning that is truly unparalleled by other systems.

11 The future of BibTeX

Again, the variety of styles is a challenge: the CSL repositories present the final user who is not a specialist in computing with +9000 styles; select the name of the needed journal or publishing house,

[10] See https://ctan.org/pkg/copac-clean

[11] Markdown – designed in 2004 by John Gruber and Aaron Swartz – is riding on a wave of popularity, also because pandoc makes the conversion to output formats a matter of a few keystrokes: LHB lists 20 manuals on Markdown, all published between 2013 and 2019. For pandoc see the online documentation (https://pandoc.org/).

and that's all. The Zotero page features also a clever system offering the user a way to find a style strictly similar to the one required by his publisher, if not already listed. BibTEX users can choose within a certain amount of choices (around 180–200), but this is ridiculous in comparison with the wealth offered by the Citation Style Language repositories. A good project would be a translation of all these styles in order for them to be usable with biblatex – or, even better, to make Citation Style Language directly digestible by biblatex. BibTEX archives + CSL formatting, thanks to pandoc, is a good example for the future.

The reason of this relative paucity of styles lies in the historical roots of LATEX and of TEX itself, originally aimed to users in the field of mathematics and informatics. The present writer, who happens to be a humanist, believes that LATEX offers e.g. to linguists, historian, and philosophers, an ideal environment to maximise productivity without wasting time and energy in trivial matters – formatting texts and, in this case, formatting bibliography: «Worrying too much about formatting and not enough about content» is the mistake that people should stop making, according to Leslie Lamport himself (Lamport, 2000, 51). A more user-friendly approach is a *desideratum* badly needed in order to make LATEX + BibTEX a real opportunity in comparison with the ubiquitous "Word + Zotero" approach. Markdown is very good for writing a paper, but the final output depends either on biber + biblatex or on pandoc-citeproc; therefore the problem of bibliography is not completely solved, as we saw above, because CSL filters do not offer the amount of details and features offered by biblatex, which on its turn has the problem of ready-to-use styles. A serious interest on the part of one or more universities (e.g. a financed research project) is needed, because the development of this complex family of software cannot rely only on the time and energy generously offered by individuals, skilled and competent as they may be.

References

Hufflen, Jean-Michael (2011). «A comparative study of methods for bibliographies». *TUGboat*, **32** (3), pp. 289–301. https://www.tug.org/TUGboat/tb32-3/tb102hufflen.pdf.

Lamport, Leslie (1994). *LATEX. A Document Preparation System. User's Guide and Reference Manual*. Addison–Wesley, Reading, Massachusetts.

— (2000). «How (La)TeX changed the face of Mathematics». *Mitteilungen der Deutschen Mathematiker-Vereinigung*, (1), pp. 49–51.

Lehman, Philipp, Philip Kime, Audrey Boruvka and Joseph Wright (2018). *The biblatex Package. Programmable Bibliographies and Citations. Version 3.12*. http://mirrors.ctan.org/macros/latex/contrib/biblatex/doc/biblatex.pdf.

McGann, Jerome (2016). *Marking Texts of Many Dimensions*, John Wiley & Sons, Ltd, Chichester, West Sussex, UK, capitolo 25, pp. 358–376. Blackwell companions to literature and culture 93.

Milanese, Guido (2015). «Zbl-build: a GUI interface for Biblatex». *ArsTEXnica*, **20**, pp. 31–34.

Mittelbach, Frank, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley (2004). *The LATEX Companion. Second edition*. Addison–Wesley Publishing Company, Reading, Massachusetts.

Valbusa, Ivan (2010). «Creare stili bibliografici con biblatex: l'esperienza del pacchetto biblatex-philosophy». *ArsTEXnica*, **9**, pp. 39–50.

— (2018). *The biblatex-philosophy bundle v1.9.8a*. http://www.ctan.org/pkg/biblatex-philosophy.

Voss, Herbert (2011). *Bibliografien mit LATEX*. Lehmanns Media, Berlin. https://books.google.it/books?id=SgNACgAAQBAJ&pg=PA228&dq=biblatex&hl=it&sa=X&ved=0CCQQ6AEwAGoVChMI-dH5o8vJxwIVR1kUCh1uxwKa.

▷ Guido Milanese
UCSC – Milano; USI – Lugano
guido dot milanese at unicatt dot it

Graphics for LaTeX users

Agostino De Marco

Abstract

This article presents the most important ways to produce technical illustrations, diagrams and plots, which are relevant to LaTeX users. Graphics is a huge subject *per se*, therefore this is by no means an exhaustive tutorial. And it should not be so since there are usually different ways to obtain an equally satisfying visual result for any given graphic design. The purpose is to stimulate readers' creativity and point them to the right direction. The article emphasizes the role of tikz for programmed graphics and of inkscape as a LaTeX-aware visual tool. A final part on scientific plots presents the package pgfplots.

Sommario

Questo articolo presenta gli strumenti più importanti per produrre illustrazioni tecniche, diagrammi e grafici, che sono rilevanti per gli utenti di LaTeX. La grafica è un argomento di per sé molto vasto, quindi questo tutorial non ha la pretesa di essere esauriente. Né dovrebbe esserlo poiché di solito ci sono più modi per ottenere un risultato visivo soddisfacente per un determinato progetto grafico. Il tentativo è di stimolare la creatività del lettore e di indirizzarlo nella direzione giusta. L'articolo sottolinea il ruolo di tikz per la grafica realizzata con codice LaTeX e di inkscape come strumento visivo capace di interfacciarsi con un sistema TeX. L'ultima parte riguarda i grafici scientifici e presenta il pacchetto pgfplots.

1 Introduction

People writing in technical professions — whether they are primarily technical communicators, engineers, scientists, or others — spend a lot of time describing technology, experiments, how things work, what a project entails, and so forth.

On-the-job technical communications often use graphics, such as the illustrations reported in Figure 1, rather than text to convey key points and information. Graphics are photographs, drawings, flowcharts, fancy tables, and other visual representations. As research shows, they play a critically important role in technical and scientific writing. Visual material convey certain kinds of information more clearly, succinctly, and forcefully than words.

One of the golden rules of traditional typography says that both the text and the accompanying visual material has to be composed to create a readable, coherent, and visually satisfying whole that works invisibly, without the awareness of the reader. Typographers and graphic designers claim that an even distribution of typeset material and graphics, with a minimum of distractions and anomalies, is aimed at producing clarity and transparency. This is even more true for scientific or technical texts, where also precision and consistency are of the utmost importance.

Authors of technical texts are required to be aware and adhere to all the typographical conventions on symbols. The most important rule in all circumstances is *consistency*. This means that a given symbol is supposed to always be presented in the same way, whether it appears in the text body, a title, a figure, a table, or a formula. A number of fairly distinct subjects exist in the matter of typographical conventions where proven typesetting rules have been established. Some examples include: (a) the correct display of units of measurement, (b) mathematical formulae, both inline and in display, (c) chemical elements and formulae, (d) numbers, (e) abbreviations. All the rules have to be applied also in visual material.

When dealing with graphics, a typical anomaly may arise when textual annotations do not match with the general design of the main document. This may happen because differences in font usage are evident or because visual signs are inappropriately crafted. Fortunately, LaTeX can be used natively to produce all sorts of visuals, to typeset the annotations of pictures and drawings, and to produce professional quality graphs. More flexible approaches are also available, with the possibility to combine the typesetting strength of LaTeX with specialized graphical software (external to the TeX system).

In the following sections we will focus on illustrations, how they are designed, how they can be generated and handled, and how their textual annotations are typeset with LaTeX. In the second part of the article we will see how scientific plots can be produced according to the same set of quality criteria.

2 Illustrations: general guidelines

The term *illustration* will refer to all kind of pictorial graphics — photographs, drawings, diagrams, and schematics. As mentioned in previous section, it is important in typography to maintain a consistency between text and graphics. When this is achieved the aesthetic result is of such a good quality that the fame of LaTeX as a tool to produce 'beautiful documents' is readily confirmed.

When it comes to producing graphics in the LaTeX world the reader is referred to the book *The LaTeX Graphics Companion* by Goosens *et al.* (2007), where many techniques can be found that let us generate, manipulate, and integrate graphics with texts. Due to several recent improvements in the TeX typesetting system, that brought the users to harness more efficiently both the features of PDF and the resources of their operating system — such as fonts installed outside TeX — , the Graphics Companion does not address some techniques that nowadays are considered standard. These rely on the program pdflatex — or on the more recent xelatex and lualatex — and on the power of the pgf package with his high-level interface tikz. One of the aims of this article is to cover these aspects.

There are many benefits coming from a careful use of visual material in technical documents. These include the following:

- Readers look for and want graphics. They gain more knowledge from communications with graphics, and remember more from communications with graphics.
- Graphics enhance a communication's visual appeal, thereby increasing the readers' concentration on its message.
- Graphics convey some kinds of information much more efficiently than prose. An example of what a reader perceives when flipping through a technical publication is shown by Figure 2. In the picture, the right-hand page contains a detailed illustration with several annotated indications. Well-crafted graphics really can say more than many lines of text.
- Graphics enable writers to convey information to readers who do not share a common language with the writers — or with each other. Graphics communicate information so effectively that they sometimes convey the entire message. An example is given by Figure 1a where the concept of Reflex in modern cameras is so evident.

Examples of visual material of all kinds are shown in the book by Harris (1996), a comprehensive illustrated reference on information graphics.

Generally speaking, when planning a communication, authors should look for places where graphics provide the best way for them to show how something looks (in drawings or photographs), explain a process (flowcharts), make detailed information readily accessible (tables), or clarify the relationship among groups of data (graphs). When the document is typeset with LaTeX, authors have a number of options to produce and handle graphics. Details of the most popular and up-to-date techniques are going to be discussed in the following sections.

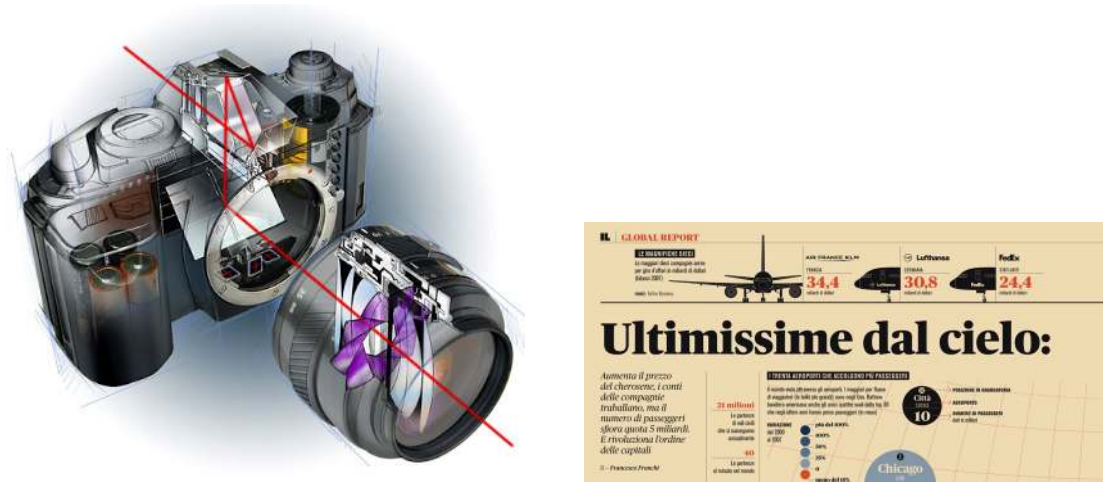
2.1 Guidelines for illustration design

When planning to include an illustration in a document one should keep in mind that, at some point, readers' attention will be going back and forth between the text and the figure, necessarily. Authors should make the effort of having the readers feel at ease during the process. Therefore, having chosen the type of graphic, it must be designed appropriately, with a special focus on usability. Graphics should have the same good qualities of author's prose, easy for readers to understand and use. Here are some general usability rules:

(i) It can be said that graphics have to be designed to support any possible readers' tasks. This is a well-known reader-centered strategy: authors should imagine their readers in the act of using their graphic. Then they should design it to support readers' efforts. In drawings or photographs for step-by step instructions, for example, one should show objects from the same angle that readers will see them when performing the actions described in the illustration and text. This principle is also valid for table design, where one should arrange columns and rows in such an order that will help readers rapidly find the particular pieces of information they are looking for.

(ii) Another important point is about readers' knowledge and expectations; these should be considered carefully by an author/illustrator. Of course, readers will find graphics useful and persuasive only if they can understand them. Some types of graphic are familiar to us all, but other types can be interpreted only by people with specialized knowledge. If one works in a field that employs specialized graphics, then these graphics only can be used when communicating with readers in that particular field, who will understand and expect them. However, when writing for a general audience, authors should consider using alternative types of graphic — or include explanations that general readers need in order to interpret special-use graphics. This kind of simplified visuals are often called 'information graphics' or 'infographics'; they include those images frequently used in presentations at formal meetings or the stylized charts and graphs used in newspapers and magazines (see, for instance, Figure 1b). Many are used for these purposes; however, for every chart, graph, map, diagram, or table used in a presentation or publication, there are thousands that are utilized in other occasions, for what are called operational purposes (Harris, 1996).

(iii) A well-known general rule-of-thumb when designing visual material is that of *seeking simplicity*. As seen in the preceding point, by simplifying their graphics authors can also make their illustrations easy to understand and use. Simplicity is especially important for graphics that will be read on a computer screen or from a projected



(a) An example of technical illustration showing the Reflex principle.

(b) A newspaper illustration. This example shows a particular kind of artwork known as 'infographics.'

Figure 1: Examples of on-the-job technical illustrations.

image; people have more difficulty reading from these media than from paper. Here are some effective strategies for keeping your graphics simple: (*a*) Include only a manageable amount of material. Sometimes, it's better to separate information into two or more graphics than to cram it all into one. (*b*) Eliminate unnecessary details. Like unnecessary words in prose, superfluous details in graphics create extra, unproductive work for readers and obscure the really important information. In many cases the elimination of extraneous detail can simplify and improve the effectiveness of a graph.

(IV) One of the most important points about illustration design is the effectiveness of textual labels. Important content should always be labelled clearly. Labels help readers locate the information in a graphic and understand what it shows. In diagrams, every part that is important to readers has to be labelled. But authors should avoid labeling other features because unnecessary labels clutter a graphic, making it difficult to understand and use. An appropriate wording should be chosen for all labels, which should be placed where they are easily seen. If necessary, a line can be drawn from the label to the item it refers to. Authors have to avoid placing a label on top of an important part in their graphic. It has to be noted that labels placed in a graphic are much easier than a key for readers to use.

(V) A special mention goes to informative titles. Titles help readers to find the graphics they are looking for and also to know what the graphics contain once they locate them. Typically, titles may go in figure captions — for example, "Figure 3. Effects of Temperature on the Strength of M312." Titles can be made brief and informative at the same time. In some cases more words can be used if they are needed in order to give readers precise information about the graphic. For this purpose LaTeX provides the `figure` environment where one can use the `\caption` macro. There are extension packages, such as `float` and `caption`, that help users customize the style of captions.

(VI) Readers might seek a specific figure whose location is not obvious from the regular table of contents. To help this process authors should provide a separate list of the figures and the pages where they can be found. Lists of figures are generated in LaTeX by the native macro `\listoffigures`.

2.2 Interplay between graphics and text

To enable graphics to achieve their potential for usability and persuasiveness, they should be carefully integrated with a communication's prose. Here are four common strategies authors can use to create a single, unified message in which their graphics and prose work harmoniously together.

(I) First of all, graphics have to be introduced in the text. When people read, they read one sentence and then the next, one paragraph and then the next, and so on. In a manuscript, when one wants to make sure that the next element the reader scans is a table or a chart rather than a sentence or a paragraph, one needs to direct people's attention from the prose to the graphic and tell them how the graphic relates to the statements they just read.

(II) Whatever kind of introduction is made in the text, it should be placed at the exact point where one wants the readers to focus their attention on the graphic. For this purpose, graphics should be placed near the point they are referenced to in the text. When readers come to a statement asking them to look at a graphic, they lift their eyes from the prose and search for the graphic. That search has to be made as short and simple as possible. Ideally, the graphic should be

Figure 2: A technical book in the hands of a reader. The right-hand page contains a full-height annotated illustration.

placed on the same page as the prose references to it. If there is not enough room, the graphic should be put on the facing page or the page that follows. If the figure is placed farther away than that (for instance, after two pages or in an appendix), the text should mention the number of the page on which the figure can be found. These strategies are handled in LaTeX by a careful positioning of floating objects in the source file and by using the cross-referencing mechanism (enanced with packages like varioref, cleveref and hyperref).

(iii) Having introduced the visual material properly, one has to state the conclusions that one wants readers to draw. One way to integrate graphics into the text is to state explicitly those conclusions. Otherwise, readers may draw conclusions that are quite different from the ones that the author has in mind.

(iv) When appropriate, graphics may include explanations, or longer annotations. Sometimes illustrations designers can help readers understand the message by incorporating explanatory statements into the figures—for instance, "Counterclockwise moments are positive by convention."

It has to be mentioned the existence of a well written section entitled "Graphics guidelines" in the pgf package documentation (Tantau, 2016), a LaTeX extension package that will be introduced later on in this paper. This material, that we encourage to read, is not only about pgf, but about general guidelines and principles concerning the creation of graphics for scientific presentations, papers, and books.

3 Drawing and annotating with native LaTeX extensions

In this section we introduce some facilities offered by LaTeX and its extension packages for producing graphic material directly in the source document.

Some of the available drawing facilities are standalone, in the sense that they rely totally on the program latex or pdflatex and do not require functionalities of other programs. Some other drawing tools, instead, rely on other programs distributed with the standard TeX system (or publicly available).

There are several drawing tools that one can pick up and use once a full TeX system is installed. But we have basicly three main facilities, which are readily listed:

(i) The package tikz, which is a high-level interface to the low-level graphics package pgf. This is considered the standard drawing facility. At the time of writing this paper (and probably for years to come) it is the most popular standalone tool for producing line graphics in the LaTeX world.[1] tikz comes with several specialized extension packages (tikz libraries). It is a very powerful package, flexible, easy to use, and stunning.

(ii) The package pstricks and its companion packages. This was the tool of choice before tikz came into the scene. pstricks is designed to embed low-level picture drawing primitives available in the PostScript language. These primitives are

[1] To learn more about pgf, see the extensive documentation on CTAN http://www.ctan.org/pkg/pgf and the collection of examples on http://www.texample.net .

exposed to the user in terms of LATEX macros so that they can work smoothly with the typesetting engine. With an an up-to-date TEX distribution pstricks can be used with pdflatex, provided that the option `-shell-escape` has been enabled.[2]

(III) The native LATEX environment picture. This environment is part of LATEX kernel and, when enhanced with the standard packages pict2e and curve2e, allows for the creation of line graphics from a number of fairly basic constructs.

The above list mentions the most important ways to produce graphics with LATEX and they will be treated in reverse order in next sections. But they are not the only possible options. The following is a list of other graphics packages and tools included in standard TEX distributions:

- The package Xy-pic. A tool which is best suited to graphs and diagrams, but has capabilities for other formats.[3]
- The package ePiX. A tool to produce mathematical figures. It creates pstricks, TikZ, or eepic macros.[4]
- The program METAPOST. Similar to the program METAFONT used to create early TEX fonts. It outputs special PostScript files that can be imported by both LATEX and PDFLATEX. it is the default drawing program used by Knuth himself. Its source code may be included into a LATEX file and, via the emp package, the METAPOST code is executed and the resulting graph is imported into the typeset document. METAPOST is now integrated in LuaTEX via the mplib library. Using LuaTEX, one can include METAPOST figures directly in the TEX/LATEX file with the luamplib package, without using any external software.[5]
- The program MetaFun. An extension to METAPOST.[6]
- The program asymptote. A descriptive vector graphics software and language for technical drawing. The language is inspired to METAPOST but with an improved syntax taken from C++.[7] LATEX users can use the package asymptote that provides the environment asy to enclose Asymptote language code into LATEX sources. See De Marco (2009) for a presentation of this powerful tool.

This latter list of tools is reported for the sake of completeness. They can be considered of secondary importance for the scope of the present article and the interested readers are referred to the suggested links and references.

Next three subsections present some selected examples of technical illustrations made, respectively, with the environments: picture (from package picture), pspicture (from package pstricks), and tikzpicture (from package tikz).

4 The standard LATEX picture environment

This section presents briefly the standard picture environment. Examples of usage of this environment are reported in Figure 3, Figure 4 and Figure 5. A picture has two main dimensions, width and height, that users can pass to the environment as arguments. In addition to picture, macros such as `\put`, `\line`, `\vector` and several other commands have arguments that expect numbers that are used as factor for `\unitlength`. The latter is a TEX length that can be set by users with the macro `\setlength` to scale their drawings as appropriate.

LATEX was born in 1984 with the native picture environment that allowed users to draw lines, vectors, circles and 'ovals' with some limitations. Special fonts were used to draw all graphic objects. This was a severe limitation, because at LATEX's birth all fonts usable with LATEX could contain only 128 glyphs. This was true with text fonts as well as with LATEX special drawing fonts.

This implied that there were only two thicknesses available for all graphic objects, and, worst of all, the line and vector slopes and the circle diameters were available in only a limited number. Line slope parameters could only be integer relatively prime numbers in the range $[-6, +6]$, while for vectors they could range only within $[-4, +4]$. This meant that only a limited set of lines and vectors could be drawn.

Filled circles (disks) were limited to diameters from 1 pt to 15 pt, while unfilled circles were limited to diameters from 1 pt to 40 pt; circles of diameter larger than 15 pt were drawn as four quarter circles, and each of these were available also for the rounded corners of 'ovals.'

This environment allowed typesetting of text by means of extensions of the `\makebox` macros, and framed text by means of `\framebox`; fine tuning of the position of the text allowed placement of any text in the precise required position and, most important of all, the fonts used were the same ones used in the text of the document as a whole. Typographically, this feature was and remains the most valuable of this built-in environment.

Leslie Lamport, in his second edition of the LATEX handbook (Lamport, 1994) fixed the syntax of a new extended picture environment, where most if not all limitations of the standard implementation could be overcome: unlimited slopes of lines and vectors, any circle radius, arbitrary line thickness also for curved lines, real third order Bézier curves, *et cetera* (see for example Figure 3).

[2] To learn more about pstricks, see the documentation on CTAN http://www.ctan.org/pkg/pstricks and the dedicated section on TUG website http://tug.org/PSTricks .

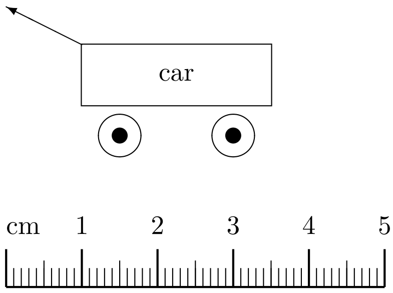
[3] http://www.tug.org/applications/Xy-pic/Xy-pic.html

[4] http://mathcs.holycross.edu/~ahwang/current/ePiX.html

[5] http://www.tug.org/metapost.html

[6] http://wiki.contextgarden.net/MetaFun

[7] https://ctan.org/pkg/asymptote

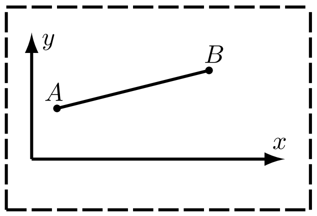


```
\setlength{\unitlength}{1mm}
% in preamble
\newcounter{cms}
% ...
\begin{picture}(50,39)
  \put(0,7){\makebox(0,0)[bl]{cm}}
  \multiput(10,7)(10,0){5}{%
    \addtocounter{cms}{1}\makebox(0,0)[
    b]{
      \arabic{cms}}}
  \put(15,20){\circle{6}}
  \put(30,20){\circle{6}}
  \put(15,20){\circle*{2}}
  \put(30,20){\circle*{2}}
  \put(10,24){\framebox(25,8){car}}
  \put(10,32){\vector(-2,1){10}}
  \multiput(1,0)(1,0){49}{\line(0,1)
    {2.5}}
  \multiput(5,0)(10,0){5}{\line(0,1)
    {3.5}}
  \thicklines
  \put(0,0){\line(1,0){50}}
  \multiput(0,0)(10,0){6}{\line(0,1)
    {5}}
\end{picture}
```

Figure 3: Lamport used this drawing in his handbook to describe his picture environment.

These extensions are nowadays incorporated in the package pict2e.

In addition to the standard pict2e package, users may rely on the curve2e package developed by Claudio Beccari.[8] This extension to pict2e enhances the syntax of the `\line` command and introduces two new commands: `\Line`, which allows the user to specify the relative x and y displacements from the current point, and `\LINE`, which has two absolute coordinates as its arguments. Similarly, `\Vector` and `\VECTOR` are defined and extend the `\vector` command. The package also defines a `\polyline` command for drawing polylines between two (minimum) or more vertices that are specified as arguments, as well as a `\Curve` command for drawing third-order Bézier curves. This macro needs a series of nodes on the curve together with the tangent at each node. Finally, curve2e introduces an `\Arc` command and some variants for drawing circular arcs of any radius and any angular aperture.



```
% in preamble
\usepackage{pict2e}
% ...
\begin{picture}(120 ,80)
  \put(30,30){\circle*{3}}
  \put(30,33){\makebox(0,0)[br]{$A$}}
  \put(90,43){\circle*{3}}
  \put(88,47){\makebox(0,0)[bl]{$B$}}
  \linethickness{1.2pt}
  \Line(30,30)(90,43)
  \put(10,10){\vector(1,0){100}}
  \put(110,14){\makebox(0,0)[b]{$x$}}
  \put(10,10){\vector(0,1){60}}
  \put(14,70){\makebox(0,0)[l]{$y$}}
  % dashed box
  \put(0,0){\dashbox{5}(120,80){}}
\end{picture}
```

Figure 4: An example showing some basic commands offered by the standard picture environment enhanced by the pict2e package.

Recently the pict2e package has incorporated some features proposed by curve2e and some drawings are made possible by simply including pict2e. Figure 4 shows an example of a very simple illustration and the related LATEX code. The example in Figure 5 demonstrates the use of tension parameters that modify the shape of a Bezier curve connecting two points.

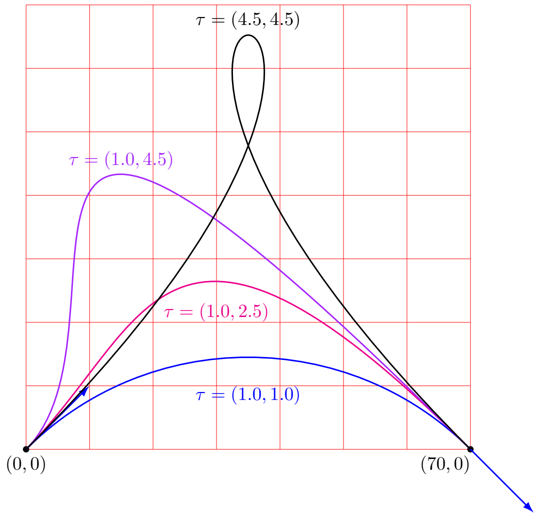
For more details on the use of the picture environment the reader might want to consult reference Beccari (2011) for a review of the available commands and for examples of graphics with this native drawing tool.

5 Using pstricks

The pstricks package is presented very briefly in this section. Explaining all the features of the package is beyond the scope of this article. For an extended treatment of the subject we encourage to consult the *LATEX Graphics Companion* (Goosens *et al.*, 2007), which contains an entire chapter on pstricks, and the book by Herbert Voß on graphics and PostScript for TEX and LATEX (Voß, 2011).

The extension package pstricks provides the environment pspicture that will contain all commands

[8] See the documentation on CTAN: https://ctan.org/pkg/curve2e.



```
% in preamble
\usepackage{pict2e}
\usepackage{curve2e}
% ...
\begin{picture}(80,70)
  \put(0,0){\GraphGrid(70,70)}
  \put( 0,0){\circle*{1}}
  \put(70,0){\circle*{1}}
  \put( 0,0){\makebox(0,-1)[ct]{$(0,0)$}}%
  \put(70,0){\makebox(0,-1)[rt]{$(70,0)$}}%

  \thicklines

  % Default tension
  \put(0,0){\color{blue}
    \Curve(0,0)<1,1>(70,0)<1,-1>
    \put( 0,0){\Vector(10,10)}
    \put(70,0){\Vector(10,-10)}
    \put(35,10){\makebox(0,0)[ct]{$\tau
    =(1.0,1.0)$}}%
  }

  \put(0,0){\color{magenta}
    \Curve(0,0)<1,1>(70,0)<1,-1;1.0,2.5>
    \put(30,23){\makebox(0,0)[ct]{$\tau
    =(1.0,2.5)$}}%
  }

  \put(0,0){\color[rgb]{0.65,0.15,1.0}
    \Curve(0,0)<1,1>(70,0)<1,-1;1.0,4.5>
    \put(15,47){\makebox(0,0)[ct]{$\tau
    =(1.0,4.5)$}}%
  }

  \Curve(0,0)<1,1>(70,0)<1,-1;4.5,4.5>
  \put(35,69){\makebox(0,0)[ct]{$\tau
    =(4.5,4.5)$}}%
\end{picture}
```

Figure 5: An example showing some basic commands offered by the standard picture environment enhanced by the curve2e package.

```
% arara: latex
% arara: dvips
% arara: ps2pdf
\documentclass[%
  border={0.6cm 0.6cm 0.6cm 0.6cm}% lbrt
  ]{standalone}

\usepackage[pdf]{pstricks}

\begin{document}
\begin{pspicture}(4,5)
  \psgrid
\end{pspicture}
\end{document}
```

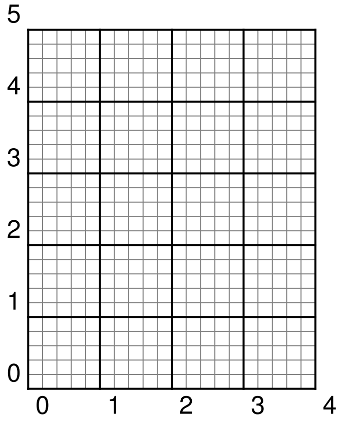


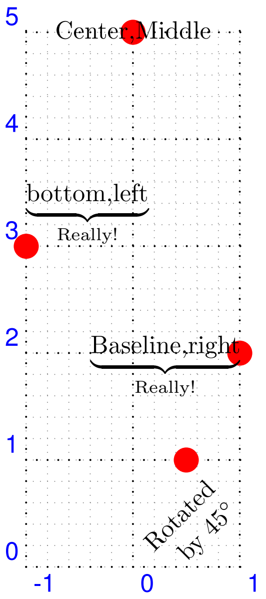
Figure 6: The basic command \psgrid offered by the pspicture environment.

necessary to produce a drawing. One important tool for drawing pictures with pstricks is the grid. It can be activated with the \psgrid macro. If no further arguments are given in the command it produces a grid with width and height as determined by the size of the enclosing pspicture. Figure 6 shows the basic commands offered by the pspicture environment. Thanks to the class standalone the output document is a PDF containing the cropped diagram. The work flow that produces the final PDF is handled by the editor texworks coupled with arara.[9]

The way pstricks works is an example of a drawing package where some of the functionalities are provided by an external program, in this case dvips. The set of macros that are collectively known as pstricks exploit the PostScript language to a great degree by writing to the output file — a .dvi file in this case — the raw PostScript code necessary to draw all of the required objects.

The most important basic geometric objects are produced by the macros \psline, \psdots, \pspolygon, \pscircle, \psellipse, \psarc,

[9] http://texdoc.net/texmf-dist/doc/support/arara/arara-usermanual.pdf



```
% arara: latex
% arara: dvips
% arara: ps2pdf
\documentclass[%
  border={0.6cm 0.6cm 0.6cm 0.6cm}% l b r t
  ]{standalone}
\usepackage[pdf]{pstricks}
\usepackage{pst-all}
\usepackage{pstricks-add}

\begin{document}
\begin{pspicture}(-1,0)(1,5)
  \psgrid[griddots=10,subgriddots=3,
    gridlabelcolor=blue](-1,0)(1,5)
  \psdots[linecolor=red,dotsize=10pt]
    (0,5)(-1,3)(1,2)(0.5,1)
  \rput(0,5){Center,Middle}
  \rput[bl](-1,3){%
    $\underbrace{\text{bottom,left}}_{\text{Really!}}$}
  \rput[Br](1,2){%
    $\underbrace{\text{Baseline,right}}_{\text{Really!}}$}
  \rput[tr]{45}(0.5,1){
    \parbox{5cm}{\flushright Rotated\\ by $45^{\circ}$}
  }
\end{pspicture}
\end{document}
```

Figure 7: Placing whatever,wherever in pspicture environment.

\pscurve, \psbezier, whose names are self-explanatory. Figure 7 shows a possible customization of the grid made by passing a number of arguments to \psgrid. The same diagram contains a few examples of how a text box can be placed on the canvas. Figure 8 demonstrates the use of \psline and \pscurve with their arguments to obtain simple lines with various line endings.

A showcase of graphics produced with pstricks and other companion packages is given by Figure 9.[10] Figure 9a represents a power balance in a form known as *Sankey diagram* and is made by coupling to pstricks the package pst-node. Figure 9b shows a three-dimensional scene produced with the package pst-solides3d. In Figure 9c the power of the package pst-plot is demonstrated by plotting the diagram of a mathematical function using 4000 connected points. In Figure 9d pst-plot is used to illustrate the construction of a hypocycloid curve. Finally, Figures 9e and 9f demonstrate the potential of package pst-3dplot.[11]

6 Using pgf/tikz

This is an introduction to drawing diagrams/pictures using the tikz package, which is built on top of pgf, a platform- and format-independent macro package for creating graphics. The pgf package is smoothly integrated with TEX and LATEX and, as a result, tikz also lets users incorporate text and mathematics in their diagrams. The tikz package also supports the beamer class, which is used for creating incremental computer presentations.

The main purpose here is to emphasize the potential of tikz and inspire a creative use of this powerful extension. The interested reader is referred to the excellent package documentation itself (Tantau, 2016) for more detailed information.

The pgf package, where 'pgf' means 'portable graphics format,' is a package for creating *inline* graphics, that is, it defines a number of TEX commands that can draw graphics within the typesetting process. Graphics objects are put into boxes and treated as normal items to be taken care of by the LATEX output routine.

As occurs with the environments picture and pspicture, when one uses pgf the graphics are *programed*, just as documents are programed when TEX is used. The package users get all the advantages of the TEX-approach to typesetting for their graphics: quick creation of simple graphics, precise positioning, the use of macros, often superior typography. But also all the disadvantages are inherited: steep learning curve, no wysiwyg, small changes require a recompilation, and the code does not really *show* how things will look like.

[10] For an extensive gallery of examples visit this link http://tug.org/PSTricks/main.cgi?file=examples .

[11] http://texdoc.net/texmf-dist/doc/generic/pst-3dplot/pst-3dplot-doc.pdf .

```
\begin{pspicture}(5,5)
  \psgrid[griddots=10,subgriddots=3,
    gridlabelcolor=blue]
  \psline{-*}(1,4)(2,4)
  \psline{-,linewidth=3pt}(3,4)(4,4)
  \psline{->,linecolor=magenta,
    linewidth=2pt}(2.5,4)(2.5,2.5)
  \pscurve{|-|}(1,2)(2.5,1)(4,2)
\end{pspicture}
```

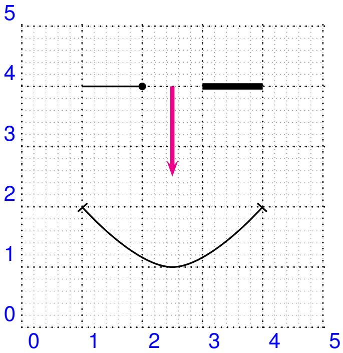


Figure 8: Lines and line endings with pstricks.

The pgf system is designed as a combination of three software layers sitting one on top of each other. The lower layer, called by the author the "system layer," is in charge of low-level tasks related to the production of the final output. In practice, the generic user will never be using the TEX macros provided by the system layer. Next, we have the "basic layer," providing a set of basic commands that allow to produce complex graphics in a much easier manner than by using the system layer directly. However, also this layer of drawing macros is not conceived to be used directly by the generic user. Finally, the pgf exposes a "frontend layer," i. e. a set of commands or a special syntax that makes using the functionalities implemented by basic layer easier. This frontend is what is called "Ti*k*Z"—hence the double possibility to name the package. The name tikz is an acronym of 'tikz ist kein Zeichenprogramm' (German for 'tikz is not a drawing program') and provides commands and environments for specifying and "drawing" graphical objects in a document.

6.1 Command \tikz and environment tikzpicture

Once \usepackage{tikz} is used in the preamble, LATEX users have two options to produce a diagram with the tikz frontend:

(*a*) The command \tikz as the following example

```
\tikz \draw (0pt,0pt) -- (20pt,6pt);
```

that yields the line , or

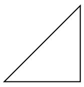
```
\tikz\fill[orange] (1ex,1ex) circle(1ex);
```

that yields the orange circle ●. Observe that the argument passed to \tikz is a string terminated by a semicolon.

(*b*) The environment tikzpicture to embed more elaborated graphic commands, as the following example

```
\begin{tikzpicture}
  \draw (0,0) -- (1,0) -- (1,1) -- cycle;
\end{tikzpicture}
```

that gives the triangle



Both the command \tikz and the environment tikzpicture can be used in running text for simple drawings. For instance, the following draws a 0.4 × 0.2 crossed rectangle: ☒.

```
The following draws a $0.4 \times 0.2$
crossed rectangle:
\begin{tikzpicture}
  \draw (0.0,0.0) rectangle (0.4,0.2);
  \draw (0.0,0.0) -- (0.4,0.2);
  \draw (0.0,0.2) -- (0.4,0.0);
\end{tikzpicture}\,.
```

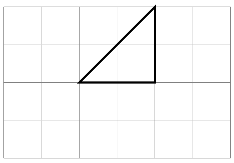
Although there is the chance to use or implement other types of frontend layers to the pgf system, the tikz frontend is by far the most popular.

6.2 Grids

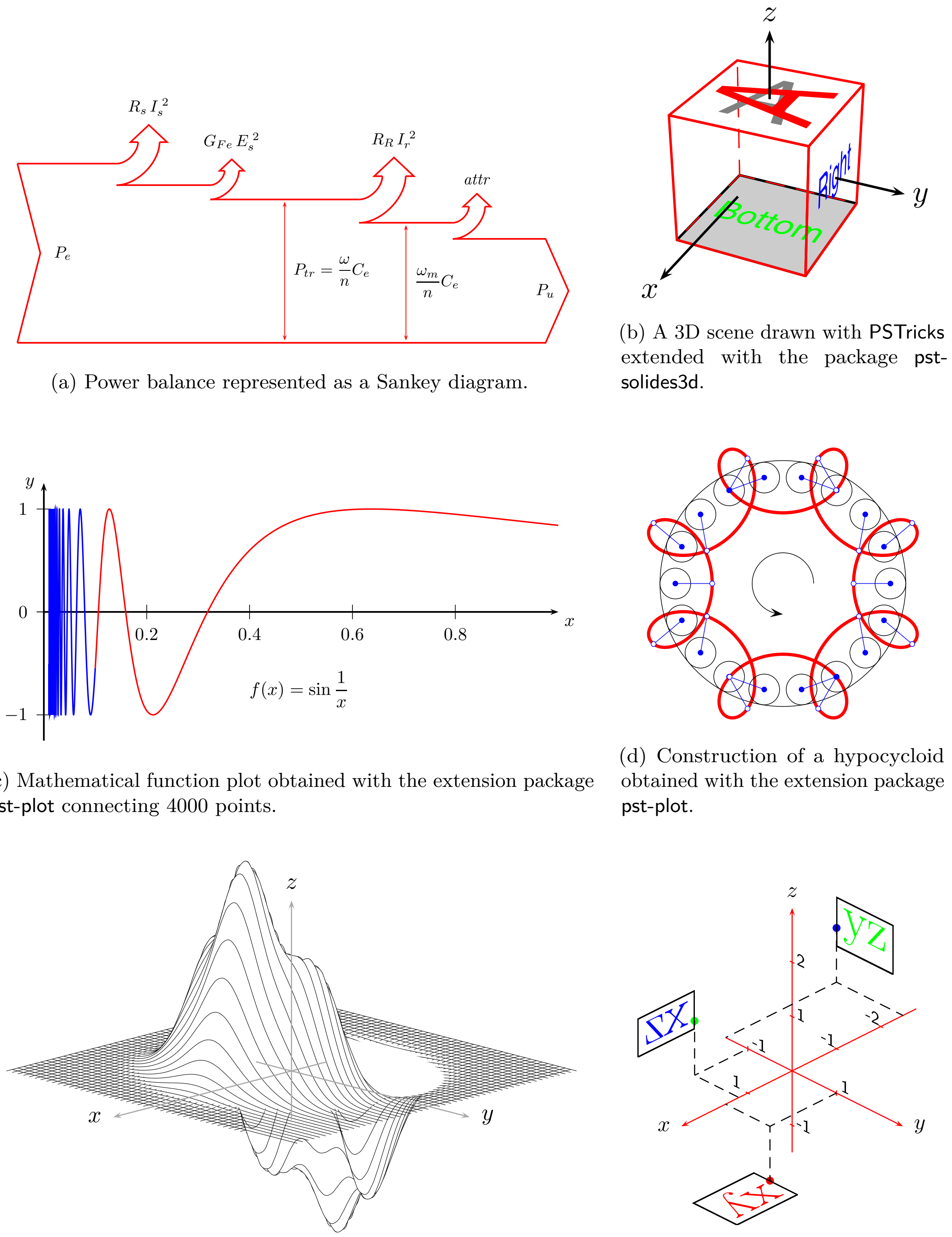
Grids, as in all programmed drawing environments, are the most important support of the trial-and-error process occurring when users develop their pictures. In tikz the simple code

```
\begin{tikzpicture}
\draw[line width=0.1pt,gray!30,step=5mm]
  (0,0) grid (3,2);
\draw[help lines]
  (0,0) grid (3,2);
\draw[thick] (1,1) -- (2,2) -- (2,1)
  -- cycle;
\end{tikzpicture}
```

draws a grid and a triangle:



This example demonstrates how to draw a basic 3×2 grid, relative to the origin. The grid consists of two superimposed grids, the coarser of which (*help lines*) is drawn on top of the other. The option gray!30 in the style of the fine grid defines the colour for the grid: one gets it by mixing 30% grey and 70% white. Of course, a grid becomes part of a picture in all cases where a scientific plot has to be represented.



(a) Power balance represented as a Sankey diagram.

(b) A 3D scene drawn with PSTricks extended with the package pst-solides3d.

(c) Mathematical function plot obtained with the extension package pst-plot connecting 4000 points.

(d) Construction of a hypocycloid obtained with the extension package pst-plot.

(e) A function $z = f(x, y)$ plotted with pst-plot3d. Reproduced from Goosens *et al.* (2007).

(f) A 3D plot obtained with the extension package pst-plot3d.

Figure 9: Examples of advanced illustrations made with PSTricks.

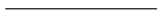
6.3 Paths

Inside a tikzpicture environment everything is drawn by starting a *path* and by *extending* the path. Paths are constructed using the \path command. In its basic form, a path is started with a coordinate that becomes the *current* coordinate of the path. Next the path is extended with other coordinates, line segments, *nodes* or other shapes. Line segments may be straight line segments or *cubic spline* segments, which are also known as *cubic splines*. Each line segment extension operation adds a line segment starting at the current coordinate and ending at another coordinate. Path extension operations may update the current coordinate.

The optional argument of the \path command is used to control if, and how the path should be drawn. Adding the option draw forces the drawing of the path. By default the path is not drawn. A semicolon indicates the end of the path. This code

```
\begin{tikzpicture}
  \path[draw] (0,0) -- (2,0);
  \path (2.5,0) -- (3,0);
\end{tikzpicture}
```

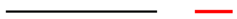
renders as follows:



The first \path command in the above tikzpicture draws a line segment from $(0,0)$ to $(2,0)$. The second \path command draws an invisible line segment. Both line segments are considered part of the picture, so the picture has a width of 3 cm. The following variant

```
\begin{tikzpicture}
  \path[draw] (0,0) -- (2,0);
  \path[draw, red, thick] (2.5,0) --
    (3,0);
\end{tikzpicture}
```

renders both lines:

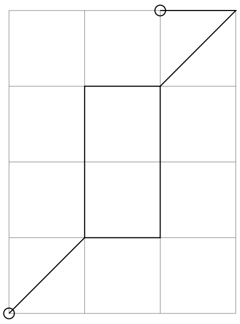


and modifies their thickness and color.

The command \draw is a shorthand for \path [draw]. The tikz package has many shorthand notations like this. The following example draws a path that starts at position $(0,0)$.

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (3,4);
  \draw (0,0) circle (2pt)
    -- (1,1) rectangle (2,3)
    -- (3,4)
    -- (2,4) circle (2pt);
\end{tikzpicture}
```

First the path is extended by adding a circle. Next the path is extended with a line segment leading to $(1,1)$. Next it is extended with a rectangle, and so on. Except for the circle extension operation, each operation changes the current position of the path. The result is:



It has to be observed that the examples given so far violate every *rule of the maintainability.* For example, what if the rectangle's size were to change, what if its position were to change, what if its colour were to change? Fortunately, tikz provides users a range of commands and techniques for maintaining their diagrams. One of the cornerstones is the ability to label nodes and coordinates and use the labels to construct other nodes and shapes. In addition the packages upports hierarchies. Parent settings may be inherited by descendants in the hierarchy.

6.4 Coordinate labels

Maintaining complex diagrams defined entirely in terms of absolute coordinates is virtually impossible. Fortunately, tikz provides many techniques that help maintain a diagram. In one of these techniques relies on the possibility to you define *coordinate labels* associated to coordinates. The resulting labels can be effectively used instead of the coordinates to build up even the most a complicated diagrams.

User defines a coordinate label by the chosen label name after the coordinate keyword. Defining coordinates this way is possible at (almost) any point in a path. Once the label of a coordinate is defined, it can be used as a coordinate. The following, which draws a crossed rectangle (⌧), demonstrates the mechanism.

```
The following, which draws a
crossed rectangle
(\begin{tikzpicture}
  \draw (0.0,0.0) coordinate(lower left)
    -- (0.4,0.2) coordinate(upper right);
  \draw (0.0,0.2) -- (0.4,0.0);
  \draw (lower left) rectangle (upper
   right);
\end{tikzpicture}), demonstrates
the mechanism.
```

A label name may contain spaces.

6.5 Types of path extensions

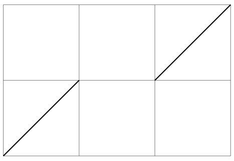
Paths are constructed by extending them. There are several different kinds of path extension operations. The majority of these extension operations modify the current coordinate, but some don't. In the remainder of this section it is therefore assumed that an extension operation modifies the current coordinate unless this is indicated otherwise. For the moment it is assumed that none of the coordinates are relative or incremental coordinates.

6.5.1 The move-to operation

The *move-to* operation is the most intuitive and adds a coordinate to the path, making it the current coordinate. The following example uses three move-to operations. The first move-to operation defines the lower left corner of the grid. The remaining move-to operations define the starts of two line segments. The code

```
\draw[help lines] (0,0) grid (3,2);
\draw (0,0) -- (1,1) % then move-to
(2,1) -- (3,2);
```

yields:

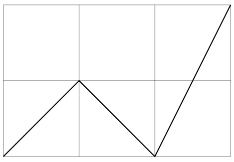


6.5.2 The line-to operation

The *line-to* operation is represented by the `--` directive and adds a straight line segment to the path. The line segment is from the current coordinate and ends in the given coordinate. The example code

```
\draw[help lines] (0,0) grid (3,2);
\draw (0,0) -- (1,1) --
  (2,0) -- (3,2);
```

yields:

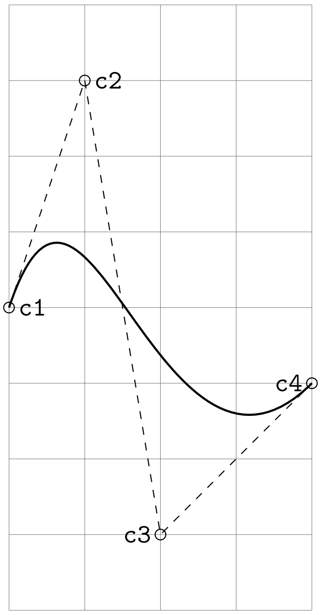


6.5.3 The curve-to operation

The *curve-to* operation is represented by the `..` directive and adds a cubic Bézier spline segment to the path. The start point of the curve is the current point of the path. The end point is last given coordinate, and the control points are all the intermediate coordinates passed to the operation. The following example code

```
% grid
\draw[help lines] (-2,-4) grid (2,4);

% define labels (nodes)
\path (-2, 0) coordinate(c1)
  (-1, 3) coordinate(c2)
  ( 0,-3) coordinate(c3)
  ( 2,-1) coordinate(c4);

% segments connecting nodes
\draw[dashed] (c1) -- (c2) -- (c3) -- (c
   4);

% control points
\draw (c1) circle (2pt)
  (c2) circle (2pt)
  (c3) circle (2pt)
  (c4) circle (2pt)
  (c1);

% the curve
\draw[thick] (c1) .. controls (c2)
  and (c3) .. (c4);

% text labels
\path
(c1) node[anchor=west] {\texttt{c1}}
(c2) node[anchor=west] {\texttt{c2}}
(c3) node[anchor=east] {\texttt{c3}}
(c4) node[anchor=east] {\texttt{c4}};
```

yields:



The above drawing demonstrates the operation. The curve starts at `c1` and ends at `c4`. The control points are given by `c2` and `c3`. The tangent of the spline segment at `c1` is equal to the tangent of the line segment `c1 -- c2`. Likewise, the tangent at `c4` is given by the tangent of the line segment `c3 -- c4`.

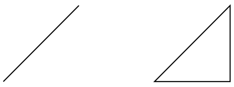
As an alternative, in the this curve-to operation the `and` can be replaced by `..` directives.

6.5.4 The cycle operation

The *cycle* operation closes the current path by adding a straight line segment from the current point to the most recent destination point of a move-to operation. The cycle operation has three applications. First it closes the path, which is required if one wishs to fill the path with a colour. Second, it connects the start and end line segments in the path. Third, it avoids the need to reference the start point of the path. The following example code

```
\draw (0,0) -- (1,1)
  (2,0) -- (3,0) --
  (3,1) -- cycle;
```

yields:

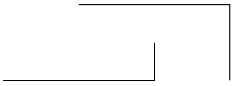


6.5.5 Connecting points with horizontal/vertical lines

This operation is equivalent to two line-to operations connecting the current coordinate and the given coordinate. It may follow the `-|` directive, that is, the first operation adds a horizontal and the second a vertical line segment. The following example code

```
\draw (0.0,0.0) -| (2.0,0.5)
  (1.0,1.0) -| (3.0,0.0);
```

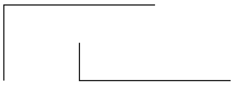
yields:



As an alternative, the operation may follow the `|-` directive. This time, however, the first operation adds a vertical and the second a horizontal line segment, as in the following example

```
\draw (0.0,0.0) |- (2.0,1.0)
  (1.0,0.5) |- (3.0,0.0);
```

resulting in:

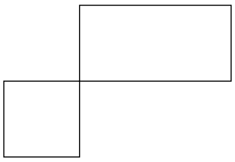


6.5.6 The rectangle operation

The rectangle operation adds a rectangle to the path. The rectangle is constructed by making the current coordinate and the given coordinate, respectively, the lower left and upper right corners of the rectangle. The following example

```
\draw (0,0) rectangle (1,1)
  rectangle (3,2);
```

yields:



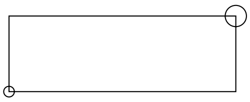
The given coordinate in the first `rectangle` operation here becomes the current coordinate in the next one.

6.5.7 The circle operation

The circle operation adds a circle to the path. The centre of the circle is given by the current coordinate of the path and its radius is the dimension passed as argument. This operation does not change the current coordinate of the path. The following example

```
\draw (0,0) circle (2pt)
  rectangle (3,1)
  circle (4pt);
```

yields:

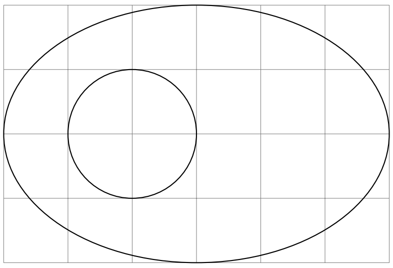


6.5.8 The ellipse operation

The ellipse operation adds an ellipse to the path. The centre of the ellipse is given by the current coordinate of the path and its semi-width and semi-height are passed as arguments. This operation does not change the current coordinate of the path. The following example

```
\begin{tikzpicture}[scale=0.85]
  \draw[help lines] (0,0) grid (6,4);
  \draw (2,2) ellipse (1cm and 1cm)
    (3,2) ellipse (3cm and 2cm);
\end{tikzpicture}
```

yields:

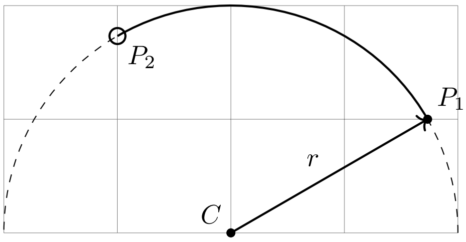


6.5.9 The arc operation

The arc operation adds an arc to the path. The arc starts at the current point, P. The user supplies two angles, α and β, and a radius r. The arc is determined by a circle of radius r. The centre of the circle, C, is determined by the equation $P = C + R \times (\cos\alpha, \sin\alpha)$. The end point of the arc is given by $P = C + R \times (\cos\beta, \sin\beta)$. The arc is drawn in counterclockwise direction from the start point to the end point, which becomes the new current coordinate of the path. The following example illustrates the construction. Only the upper half of the circle is drawn. The resulting arc is drawn with a continuous line. The code

```
\begin{tikzpicture}[scale=1.5]
\draw[help lines] (0,0) grid (4,2);
\draw[dashed] (4,0) coordinate(p0)
  arc (0:180:2cm);
\draw[fill=black] (2,0) coordinate(c)
  circle(1pt)
  node[anchor=south east] {$C$};
\path (p0) arc (0:30:2cm)
  coordinate(p30);
\draw[fill=black] (p30) circle(1pt)
  node[anchor=south west] {$P_1$};
\draw[thick] (p30) arc (30:120:2cm)
  coordinate(p120)
  circle (2pt)
  node[anchor=north west] {$P_2$};
\draw[->,thick] (c) --
  node[anchor=south east] {$r$} (p30);
\end{tikzpicture}
```

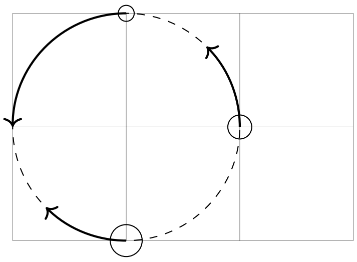
yields:



Further examples of arcs are drawn with the following code

```
\begin{tikzpicture}[scale=1.5]
  \draw[help lines] (0,0) grid (3,2);
  \draw[dashed] (1,1) circle (1cm);
  \draw (1,2) coordinate(a) circle (2pt)
    (2,1) coordinate(b) circle (3pt)
    (1,0) coordinate(c) circle (4pt);
  \draw[->,thick] (a) arc (90:180:1cm);
  \draw[->,thick] (b) arc (0:45:1cm);
  \draw[->,thick] (c) arc (270:225:1cm);
\end{tikzpicture}
```

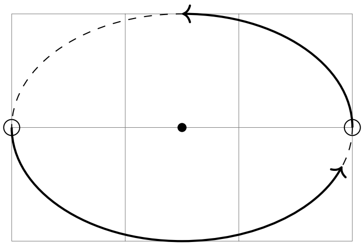
resulting in:



The arc operation can be performed along an ellipse as well. It adds an ellipse segment to the path. The construction of the ellipse segment is similar to the construction of the arc segment. In this case, instead of passing the radius, the user must provide the half width and the half height of the ellipse. The following code

```
\begin{tikzpicture}[scale=1.5]
  \draw[help lines] (0,-1) grid (3,1);
  \draw[dashed] (1.5,0) circle (1.5cm and
     1cm);
  \draw[fill=black] (1.5,0) coordinate(c)
     circle(1pt);
  \draw (3,0) coordinate(a) circle (2pt);
  \draw (0,0) coordinate(b) circle (2pt);
  \draw[->,thick] (a) arc (0:90:1.5cm and
     1cm);
  \draw[->,thick] (b) arc (180:340:1.5cm
    and 1cm);
\end{tikzpicture}
```

demonstrates the elliptical arcs:



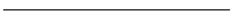
6.6 Actions on paths

Most of the examples shown so far are conceived to emphasize the default path style. This may not always be what users want. For example, one may want to draw a line in a certain colour, change the default line width, fill a shape with a colour, and so on. In tikz terminology this is achieved with *path actions*, which are operations acting on an existing path. The user first constructs the path and then apply the action. At the basic level the command `\draw` is defined in terms of an action on a path: the action results in the path being drawn. As pointed out before `\draw` is a shorthand for `\path[draw]`.

The following are some other shorthand commands that are defined in terms of path actions inside the tikzpicture environment.

`\draw` Shorthand for `\path[draw]`.
Example:

```
\draw (0,0) -- (3,0);
```



`\fill` Shorthand for `\path[fill]`.
Example:

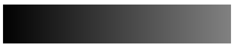
```
\fill[gray!30] (0,0) rectangle (3,0.5);
```

`\filldraw` Shorthand for `\path[filldraw]`.
Example:

```
\filldraw[fill=gray!30,draw=black,thick]
  (0,0) rectangle (3,0.5);
```

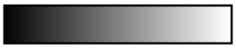
`\shade` Shorthand for `\path[shade]`. Example:

```
\shade[left color=black,right color=gray]
  (0,0) rectangle (3,0.5);
```



`\shadedraw` Shorthand for `\path[shadedraw]`. Example:

```
\shadedraw[left color=black,right color=
    white,thick]
  (0,0) rectangle (3,0.5);
```

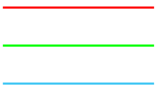


6.7 Colour

The tikz package knows several colours. Some colours are inherited from the xcolor package.[12] There are several techniques to define a new name for a colour. To learn more, the reader is referred to the documentation of xcolor.

Some path actions also let users define a colour. For example, one may draw a path with the given colour. There are different ways to control the colour. The option `color` determines the colour for drawing and filling, and the colour of text in nodes. (Nodes are explained later on in this section.) One may set the colour of the whole tikzpicture or set the colour of a given path action. Setting the colour of the whole picture is done by passing a `color` option to the environment. Setting the colour of a path action is done by passing the option to the `\path` command (or derived shorthand commands). The following is an example that draws three lines: one in red, one in green, and one in 60% cyan and 40% white.

```
\begin{tikzpicture}[thick,color=red]
  \draw (0,2) -- (2,2);
  \draw[color=green] (0,1.5) -- (2,1.5);
  \draw[color=cyan!60]
    (0,1) -- (2,1);
\end{tikzpicture}
```

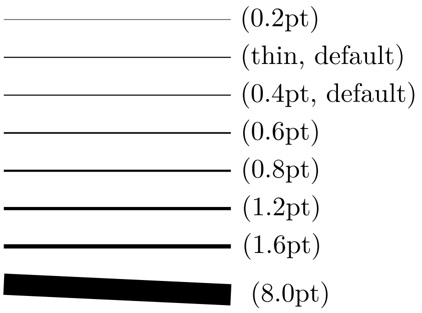


It is usually possible to omit the `color=` part when one specifies colour options.

6.8 Line width

In tikz there are several path actions affecting the line style, including the style that determines the line width, the line cap, and the line join. The following code provide some examples.

```
\draw[very thin] (0,3.5) -- (3,3.5)
  node[anchor=west] {(0.2pt)};
\draw[thin] (0,3) -- (3,3)
  node[anchor=west] {(thin, default)};
\draw[line width=0.4pt] (0,2.5) --
    (3,2.5)
  node[anchor=west] {(0.4pt, default)};
\draw[semithick] (0,2) -- (3,2)
  node[anchor=west] {(0.6pt)};
\draw[thick] (0,1.5) -- (3,1.5)
  node[anchor=west] {(0.8pt)};
\draw[very thick] (0,1.0) -- (3,1.0)
  node[anchor=west] {(1.2pt)};
\draw[ultra thick] (0,0.5) -- (3,0.5)
  node[anchor=west] {(1.6pt)};
\draw[line width=8pt] (0,0) -- (3,-4pt)
  node[anchor=west] {(8.0pt)};
```

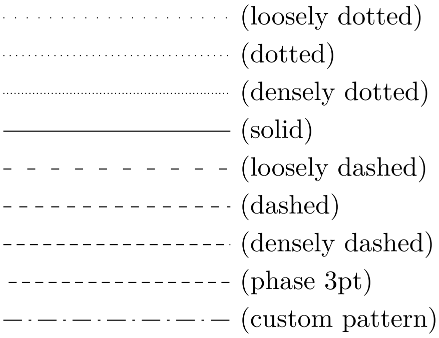


6.9 Dash patterns

The drawing of lines inb tikz also depends on the *dash pattern* and *dash phase* settings. The dash pattern determines a basic pattern for the line that is repeated cyclicly. The dash phase shifts the dash pattern. By default the dash pattern is `solid`. The following shows the relevant path actions that affect dash patterns.

```
\draw[loosely dotted] (0,3.5) -- (3,3.5)
  node[anchor=west] {(loosely dotted)};
\draw[dotted] (0,3) -- (3,3)
  node[anchor=west] {(dotted)};
\draw[densely dotted] (0,2.5) -- (3,2.5)
  node[anchor=west] {(densely dotted)};
\draw[solid] (0,2.0) -- (3,2.0)
  node[anchor=west] {(solid)};
\draw[loosely dashed] (0,1.5) -- (3,1.5)
  node[anchor=west] {(loosely dashed)};
\draw[dashed] (0,1.0) -- (3,1.0)
  node[anchor=west] {(dashed)};
\draw[densely dashed] (0,0.5) -- (3,0.5)
  node[anchor=west] {(densely dashed)};
\draw[densely dashed,
  dash phase=3pt] (0,0.0) -- (3,0.0)
  node[anchor=west] {(phase 3pt)};
\draw[dash pattern=on 7pt off 2.5pt
  on 1pt off 2.5pt] (0,-0.5) -- (3,-0.5)
  node[anchor=west] {(custom pattern)};
```

[12] http://texdoc.net/texmf-dist/doc/latex/xcolor/xcolor.pdf



6.10 Predefined styles

Hard-coding a line width or a dash pattern command is not always a good idea. It is usually better to define a style for a certain line width, for a dash style, or a combination of the two. The advantages of doing this are that you only have to define the style once and can use it several times. Using styles gives you a consistent appearance for the resulting lines, and if you want to make a global change to the style then you only have to make one change in your LaTeX file. Later on in this section it will be explained how users can define their own styles.

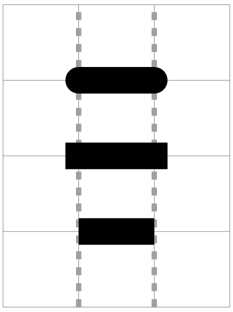
Several previous examples given so far make use of some predefined line width and dash pattern styles. The default line width is `thin`. The default dash pattern is `solid`.

6.11 Line caps and joins

The drawing of a path depends on several parameters. The *line cap* determines how lines start and end. The *line join* determines how line segments are joined.

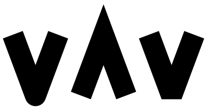
The following examples demonstrates different line cap types.

```
\begin{tikzpicture}[line width=8pt]
  \draw[help lines] (0,0) grid (3,4);
  \draw[line width=2pt,dashed,gray!75]
    (1,0) -- (1,4) (2,0) -- (2,4);
  \draw[line cap=round] (1,3) -- (2,3);
  \draw[line cap=rect] (1,2) -- (2,2);
  \draw[line cap=butt] (1,1) -- (2,1);
\end{tikzpicture}
```



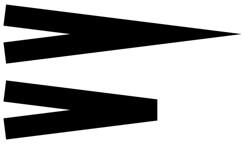
The following examples demonstrates different line join types.

```
\begin{tikzpicture}[line width=8pt]
  \draw[line join=round]
    (0.0,.8)--(0.3,.0)--(0.6,.8);
  \draw[line join=miter]
    (0.9,.0)--(1.2,.8)--(1.5,.0);
  \draw[line join=bevel]
    (1.8,.8)--(2.1,.0)--(2.4,.8);
\end{tikzpicture}
```



To avoid sharp-angled miter joins that protrude too far beyond the joining point, tikz provides the control option `miter limit`. It poses a limit on how far the miter join may protrude the joining point. If the join protrudes beyond the limit then the join style is changed to `bevel`. The limit is equal to a fraction of the line width. An example of use is the following.

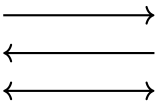
```
\begin{tikzpicture}
  [line width=8pt,line join=miter]
  \draw (0,0) -- (0.25,2) -- (0.5,0);
  \draw[miter limit=8]
    (1,0) -- (1.25,2) -- (1.5,0);
\end{tikzpicture}
```



6.12 Arrows

Arrows are also drawn using path actions. The following example demonstrates some common ways to how to draw them.

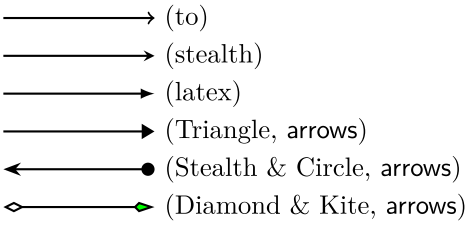
```
\begin{tikzpicture}[thick]
  \draw[->] (0,1.0) -- (2,1.0);
  \draw[<-] (0,0.5) -- (2,0.5);
  \draw[<->] (0,0.0) -- (2,0.0);
\end{tikzpicture}
```



Several arrow head styles are available besides the default one shown above. Some of the styles are provided by the tikz extension library arrows.meta. The following code demonstrates a small selection of arrow types.

```
% in preamble
\usepackage{tikz}
\usetikzlibrary{arrows.meta}

\begin{tikzpicture}[thick]
  \draw[>=to,->] (0,1.0) -- (2,1.0)
```

```
    node[anchor=west] {(to)};
  \draw[>=stealth,->] (0,0.5) -- (2,0.5)
    node[anchor=west] {(stealth)};
  \draw[>=latex,->] (0,0.0) -- (2,0.0)
    node[anchor=west] {(latex)};
 \draw[>=Triangle,->] (0,-0.5) --
    (2,-0.5)
    node[anchor=west] {(Triangle, arrows)
    };
 \draw[Stealth-Circle] (0,-1.0) --
    (2,-1.0)
    node[anchor=west] {(Stealth \& Circle
    , arrows)};
 \draw[{Diamond[open]}-{Kite[fill=green
    ]}] (0,-1.5) -- (2,-1.5)
    node[anchor=west] {(Diamond \& Kite,
    arrows)};
```



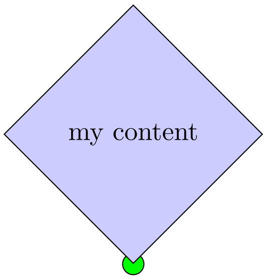
6.13 Nodes and node labels

Diagrams with lines only are rare. Usually, they also contain text, math, or both. Fortunately, tikz has a mechanism for adding text, math, and other material to paths. This is done with the *node path extension operation.*

The node path extension operation allows to place a given content (delimited by curly brackets) at the current position in the path using some given options, and associates a label to the node. Each node added to a path has an *outer shape.* The outer shape is only drawn if draw is part of the options. The default node shape is a rectangle but other shapes are also defined.

The following example draws a circle at position $(1, 0)$ and at the same time places a diamond-shaped node at the current point on the path. The node contains the words 'my content' and shape boundaries have a distance of 10 pt from the text. The fill operation is applied both to the circle and to the diamond, giving them, respectively, a green and a light blue background.

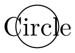
```
% in preamble
\usepackage{tikz}
\usetikzlibrary{shapes.geometric}

\draw (0,1) % current position.
  [fill=green] % options for circle
  circle (4pt) % draw shape circle
  node[anchor=south, % node options
    diamond,
    fill=blue!20,
    inner sep=10pt,draw]
    (c) % node label
    {my content}; % content
```



When adding a node to a path, it is not mandatory to have a label and a set of options. In fact, the simple code

```
\draw (0,1) circle (8pt) node {Circle};
```

just draws a circle and puts the word 'Circle' on top of it:

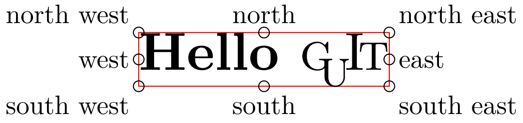


Observe that the default behaviour puts the word's center at the current coordinate.

When a node receives a label, ⟨ *label* ⟩, then usually the additional labels ⟨ *label* ⟩.center, ⟨ *label* ⟩.north, ⟨ *label* ⟩.north east, ..., and ⟨ *label* ⟩.north west are also defined. The positions of these labels correspond to their names, so ⟨ *label* ⟩.north is to the north of the node having label ⟨ *label* ⟩. This holds for the most common node shapes. Next example involves all these auxiliary labels, except for ⟨ *label* ⟩.center. The option anchor in the example is a way to override the node's default insertion point.

```
\begin{tikzpicture}
  \draw (0,0)
    node (hello)
      [scale=2.0,
      inner sep=0pt,outer sep=0pt,
      draw=red]
      {\fbox{\textbf{Hello \GuIT}}};
  \draw (hello.north east) circle (2pt)
    node[anchor=south west] {north east};
  \draw (hello.north) circle (2pt)
    node[anchor=south] {north};
  \draw (hello.north west) circle (2pt)
    node[anchor=south east] {north west};
  \draw (hello.west) circle (2pt)
    node[anchor=east] {west};
  \draw (hello.south west) circle (2pt)
    node[anchor=north east] {south west};
  \draw (hello.south) circle (2pt)
    node[anchor=north] {south};
  \draw (hello.south east) circle (2pt)
    node[anchor=north west] {south east};
  \draw (hello.east) circle (2pt)
```

```
    node[anchor=west] {east};
\end{tikzpicture}
```



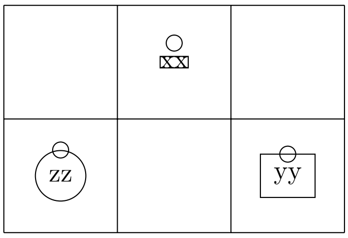
Observe in the above example that the options `inner sep` and `outer sep` are both set to 0 pt. This means that the default rectangular shape containing the words 'Hello GUIT' is the actual bounding box of the text and that the anchor points are precisely located at the boundary of the box. A nonzero `inner sep` (*inner separation*) makes the shape larger than the actual bounding box. A nonzero `outer sep` (*outer separation*) offsets the anchor points outwards of the given dimension.

6.14 Predefined node shapes

Nodes have a shape/style and content. The default node shape is rectangular but `tikz` also predefines the shapes `coordinate`, `rectangle`, `circle`, and `ellipse`. The option `shape=`⟨*shape*⟩ determines the node shape.

The following example shows some of the different node shape options and low-level control.

```
\begin{tikzpicture}[scale=1.5]
  \draw (0,0) grid (3,2);
  \draw (1.5,1.5)
    node (a)
      [draw,inner sep=0pt,outer sep=5pt]
      {xx};
  \draw (2.5,0.5)
    node (b)
      [draw,inner sep=5pt,outer sep=0pt]
      {yy};
  \draw (0.5,0.5)
    node(c)
    [draw,shape=circle] {zz};
  \draw (a.north) circle (2pt);
  \draw (b.north) circle (2pt);
  \draw (c.north) circle (2pt);
\end{tikzpicture}
```

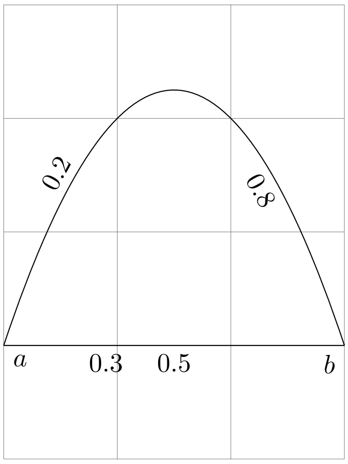


The difference in the inner separations of the rectangular nodes manifests itself in different sizes for the rectangular shapes. Differences in the outer separations result in different distances of labels such as `north`. The higher the outer separation of a node, the further its north label is away from its rectangular shape.

6.15 Node placement

Several node options exist that permit a low-level control of all graphical aspects. Here we show some of these node options with an example.

```
\begin{tikzpicture}[scale=1.5]
  \draw[help lines] (0,0) grid (3,4);
  \draw (0,1) coordinate(a)
    node[anchor=north west] {$a$}
    -- (3,1) coordinate(b)
    node[anchor=north east] {$b$}
    node[pos=0.3,anchor=north] {$0.3$}
    node[pos=0.5,anchor=north] {$0.5$}
    (a) .. controls (1,4) and (2,4) .. (b
    )
    node[pos=0.2,sloped,anchor=south] {$
    0.2$}
    node[pos=0.8,sloped,anchor=north] {$
    0.8$};
\end{tikzpicture}
```

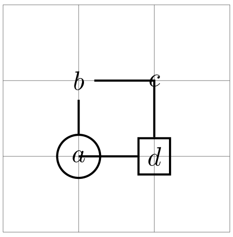


Notice that several nodes can be placed with `pos` options for the same path segment.

6.16 Connecting nodes

The `tikz` package is well-behaved. It will not cross lines unless user says so. This includes the crossing of borderlines of node shapes. For example, let us assume the user created two nodes. One of them is a circle, which is labelled `c`, and the other is a rectangle, which is labelled `r`. When user draws a line using the command `\draw (c) -- (r);` then the resulting line segment will not join the centres of the two nodes. The actual line segment will be shorter because the line segment starts at the circle shape and ends at the rectangle shape. In most cases this is the desired behaviour. If one needs a line between the centres then `.center` notation must be used. The following code provides an example.

```
\begin{tikzpicture}[thick]
  \draw[help lines] (0,0) grid (3,3);
  \path (1,1) node(a)[draw,shape=circle]
    {$a$};
  \path (1,2) node(b)[shape=rectangle] {$
    b$};
  \path (2,2) node(c)[shape=circle] {$c$
    };
  \path (2,1) node(d)[draw,shape=
    rectangle] {$d$};
  \draw (a) -- (b) -- (c.center) -- (d)
    -- (a.center);
\end{tikzpicture}
```



6.17 Coordinate systems

Specifying coordinates is the key to effective, efficient, and maintainable picture creation. Coordinates may be specified in different ways each coming with its own specific coordinate system. Within a coordinate system you specify coordinates using *explicit* or *implicit* notation.

Explicit — Explicit coordinate specifications are verbose. To specify a coordinate, users write `(⟨system⟩ cs: ⟨coord⟩)`, where `⟨system⟩` is the name of the coordinate system and where `⟨coord⟩` is a coordinate whose syntax depends on `⟨system⟩`. For example, to specify the point having x-coordinate `⟨x⟩` and y-coordinate `⟨y⟩` in the canvas coordinate system one writes `(canvas cs:x=⟨x⟩, y=⟨y⟩)`.

Implicit — Implicit coordinates specifications are shorter than explicit coordinate specifications. Users specify coordinates using some coordinate system-specific notation inside parentheses. Most examples so far have used the implicit notation for the canvas coordinate system.

Canvas coordinate system — The most widely used coordinate system is the `canvas` coordinate system. It defines coordinates in terms of a horizontal and a vertical offset relative to the origin. The implicit notation `(⟨x⟩, ⟨y⟩)` is the point with x-coordinate `⟨x⟩` and y-coordinate `⟨y⟩`.

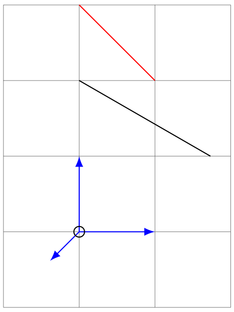
Xyz coordinate system — The `xyz` coordinate system defines coordinates in terms of a linear combination of an x-, a y-, and a z-vector. By default, the x-vector points 1 cm to the right, the y-vector points 1 cm up, and the z-vector points to $(-\sqrt{2}/2, -\sqrt{2}/2)$. However, these default settings can be changed. The implicit notation `(⟨x⟩, ⟨y⟩, ⟨z⟩)` is used to define the point at `⟨x⟩` times the x-vector plus `⟨y⟩` times the y-vector plus `⟨z⟩` times the z-vector.

Polar coordinate system — The canvas polar coordinate system defines coordinates in terms of an angle and a radius. The implicit notation `(`α`:`r`)` corresponds to the point $(r\cos\alpha, r\sin\alpha)$. Angles in this coordinate system, as all angles in `tikz`, should be supplied in degrees.

Node coordinate system — The node coordinate system defines coordinates in terms of a label of a node or coordinate. The implicit notation `(⟨label⟩)` is the position of the node or coordinate that was given the label `⟨label⟩`.

The following example demonstrates the previous four coordinate systems in action. The optional argument of the `tikzpicture` sets the arrow head style to the predefined style named `latex`.

```
\begin{tikzpicture}[>=latex]
  \draw[help lines] (-1,-1) grid (2,3);
  \draw[red] (canvas cs:x=1cm,y=2cm) --
    (0,3);
  \draw[blue,->] (0,0) -- (xyz cs:x=1,y
    =0,z=0);
  \draw[blue,->] (0,0) -- (0,1,0);
  \draw[blue,->] (0,0) -- (0,0,1);
  \draw (canvas polar cs:radius=2cm,angle
    =30)
    -- (90:2);
  \path (0,0) coordinate (origin);
  \draw (origin) circle (2pt);
\end{tikzpicture}
```



Users can freely mix the coordinate systems. For example `\draw (0,0) -- (0,1);` and `\draw (0,0) -- (90:1);` are equivalent.

6.18 Relative and incremental coordinates

Specifying diagrams in terms of absolute coordinates is cumbersome and prone to errors. What is worse, diagrams defined in terms of absolute coordinates are difficult to maintain. For example, changing the position of an n-agon that is defined in terms of absolute coordinates requires changing n coordinates. Fortunately, `tikz` provides a coordinate computation mechanism based on previously defined coordinates. Used intelligently, this reduces the maintenance costs of diagrams.

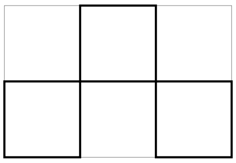
Relative and *incremental* coordinates are computed from the current coordinate in a path. The first doesn't change the current coordinate whereas the second does change it.

Relative coordinate — A relative coordinate constructs a new coordinate at an offset from the current coordinate without changing the current coordinate. The notation +⟨ *offset* ⟩ specifies the relative coordinate that is located at offset ⟨ *offset* ⟩ from the current coordinate.

Incremental coordinate — An incremental coordinate also constructs a new coordinate at an offset from the current coordinate. This time, however, the new coordinate becomes the current coordinate. One uses the implicit notation ++⟨ *offset* ⟩ for incremental coordinates.

The following example draws three squares. The first square is drawn with absolute coordinates, the second with relative coordinates, and the last with incremental coordinates.

```
\begin{tikzpicture}[thick]
  \draw[help lines] (0,0) grid +(3,2);
  \draw (0,0) -- (+1,0) --
    (1,1) -- (+0,1) -- cycle;
  \draw (1,1) -- +(+1,0) --
    +(1,1) -- +(+0,1) -- cycle;
  \draw (2,0) -- ++(+1,0) --
    ++(0,1) -- ++(-1,0) -- cycle;
\end{tikzpicture}
```



Clearly, the relative and incremental coordinates should be preferred because they improve the maintenance of the picture. For example, moving the first square requires changing four coordinates, whereas moving the second or third square requires changing only the start coordinate. The relative coordinate in the grid also improves the maintainability.

6.18.1 Complex coordinate calculations

Finally, tikz offers complex coordinate calculations. However, these calculations are only available if the tikz extension library calc is loaded in the preamble.

Generally, coordinate computations based on previously defined points are enclosed in the special syntax

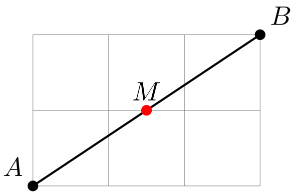
($ ⟨*coordinate modifiers*⟩ $)

where *coordinate modifiers* are a set of possible constructs that usually manipulate two existing points to produce a new one.

The following examples present several coordinate computations involving *distance modifiers*.

The code

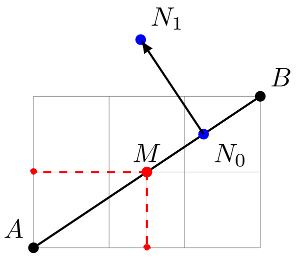
```
% in preamble
\usetikzlibrary{calc}

\begin{tikzpicture}[thick]
  \draw[help lines] (0,0) grid +(3,2);
  \path (0,0) coordinate (A)
    [fill]circle (2pt) node[anchor=south
    east] {$A$};
  \path +(3,2) coordinate (B)
    [fill]circle (2pt) node[anchor=south
    west] {$B$};
  \draw (A) -- (B);
  \path ($(A)!0.5!(B)$) coordinate (M)
    [fill=red]circle (2pt) node[anchor=
    south] {$M$};
\end{tikzpicture}
```



calculates the halfway point (M) between two coordinates, (A) and (B), using the special syntax ($(A)!0.5!(B)$).

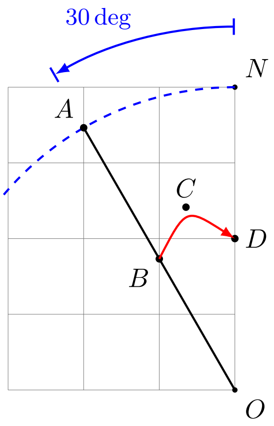
The last example can be elaborated further by extracting the x- and y-coordinate of (M) using the special path operation \let, see the package documentation (Tantau, 2016) for a detailed explanation of this handy feature. Finally, a point N_0 is taken on segment AB at $\frac{3}{4}\,|AB|$ from A and a point N_1 is calculated such that segment N_0N_1 is normal to AB and $|N_0N_1| = 1.5\,\mathrm{cm}$.

```
% in preamble
\usetikzlibrary{calc}

\begin{tikzpicture}[thick,>=latex]
  \draw[help lines] (0,0) grid +(3,2);
  \path (0,0) coordinate (A)
    [fill]circle (2pt) node[anchor=south
    east] {$A$};
  \path +(3,2) coordinate (B)
    [fill]circle (2pt) node[anchor=south
    west] {$B$};
  \draw (A) -- (B);
  \path ($(A)!0.5!(B)$) coordinate (M)
    [fill=red]circle (2pt) node[anchor=
    south] {$M$};
  % extract M.x
  \draw[dashed,red]
    % point register <-- M coordinates
    let \p{M}=(M) in
    (M) -- (\x{M},0) % extract M.x
      [fill=red]circle(1pt)
    (M) -- (0,\y{M}) % extract M.y
      [fill=red]circle(1pt);
  % point N0
  \path ($(A)!0.75!(B)$) coordinate (N0)
    [fill=blue]circle (2pt) node[anchor=
    north west] {$N_0$};
  % point N1: N0--N1 normal to N0--B
  \path ($(N0)!1.5cm!90:(B)$) coordinate
    (N1);
  \path (N1) [fill=blue]circle (2pt) node
    [anchor=south west] {$N_1$};
  \draw[->] (N0) -- (N1);
\end{tikzpicture}
```



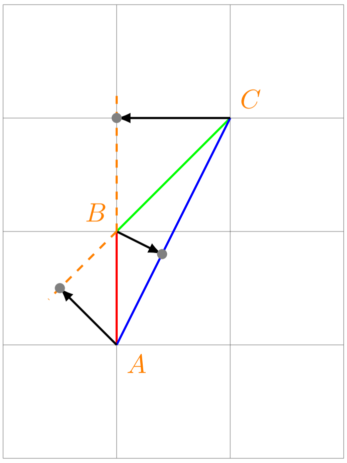
The following example demonstrates more computations with *partway* and distance modifiers.

```
% in preamble
\usetikzlibrary{calc}

\begin{tikzpicture}[thick,>=latex]
  \draw[help lines] (-3,0) grid +(3,4);
  % define origin O
  \path (0,0) coordinate (O)
    [fill] circle (1pt)
      node[anchor=north west] {$O$};
  % define point N
  \path (0,4) coordinate (N)
    [fill] circle (1pt)
      node[anchor=south west] {$N$};
  % point computation helpers
  \draw[dashed,blue] (0,4) arc (90:140:4)
    ;
  \draw[|->|,blue] (0,4.8) arc
    (90:120:4.8)
    node[pos=0.5,anchor=south east]
      {\small 30\,deg};
  % compute A, B, C, D
  \draw (O)
    % connect the origin
    -- % with next point
    % define A:
    %  on segment ON,
    %  at N,
    %  then rotate of 30deg about O
    ($(O)!1.0!30:(N)$)
      coordinate (A)
      [fill] circle (1pt)
      node[anchor=south east] {$A$}
    % define B:
    %  on segment OA,
    %  at 2cm from O
    ($(O)!2.0cm! (A)$)
      coordinate (B)
      [fill] circle (1pt)
      node[anchor=north east] {$B$}
    % define C:
    %  on segment OA,
    %  at 2.5cm from O
    %  then rotate of -15deg about O
    ($(O)!2.5cm!-15:(A)$)
      coordinate (C)
      [fill] circle (1pt)
      node[anchor=south] {$C$}
    % define D:
    %  on segment OA,
    %  at 2cm from O,
    %  then rotate of -30deg about O
    ($(O)!2cm!-30:(A)$)
      coordinate (D)
      [fill] circle (1pt)
      node[anchor=west] {$D$};
  % draw a bezier
  \draw[-latex,red]
    (B) .. controls (C) .. (D);
\end{tikzpicture}
```



Finally, next example demonstrate coordinate computations with projection modifiers.

```
% in preamble
\usetikzlibrary{calc}

\begin{tikzpicture}[scale=1.5,
  thick,>=latex,line join=round]
  \draw[help lines] (0,0) grid +(3,4);
  \draw[red] (1,1) coordinate (A)
    node[anchor=north west,orange] {$A$}
    % segment AB
    -- (1,2) coordinate (B)
      node[anchor=south east,orange] {$B$
    };
  \draw[green] (B)
    % segment BC
    -- (2,3) coordinate (C)
      node[anchor=south west,orange] {$C$
    };
  % segment CA
  \draw[blue] (C) -- (A);
  % connect points with their projections
  % on opposite segments
  % B on segment AC
  \draw[->] (B) -- ($(A)!(B)!(C)$)
    coordinate (Bp);
  \draw[fill,gray] (Bp) circle (1pt);
  % C on segment BA
  \draw[->] (C) --
    ($(B)!(C)!(A)$) coordinate (Cp);
  % construction helpers
  \draw[dashed,orange] (B)
    -- ($(B)!1.2!(Cp)$);
  \draw[fill,gray] (Cp) circle (1pt);
  % A on segment CB
  \draw[->] (A) --
    ($(C)!(A)!(B)$) coordinate (Ap);
  % construction helpers
  \draw[dashed,orange] (B)
    -- ($(B)!1.2!(Ap)$);
  \draw[fill,gray] (Ap) circle (1pt);
\end{tikzpicture}
```



In the last drawing three vertices of a triangle ABC are first defined, annotated, and connected. Successively, some helper lines and dots are represented as aids to the reader. Finally, each vertex V is projected onto the opposite segment S_1S_2 according to the sintax `($(`S_1`)!(`V`)!(`S_2`)$)`.

6.19 What else?

Making graphics with pgf is a huge subject. Therefore, this section does not claim to serve as an exhaustive tutorial on programmed illustrations with tikz. There are several aspects that have been left out of this presentation to save space and remain on the essential commands and options.

For a comprehensive explanation of tikz styles, scopes, options, advanced path operation, customization possibilities, and extension libraries the interested readers are referred to the excellent user guide (Tantau, 2016) and to the book *LATEX and Friends* (van Dongen, 2012).

All tikz examples given in previous subsections are viewable on Overleaf website.[13] These are provided mainly to facilitate new users in their further explorations.

6.20 Advanced examples

In this final subsection on tikz three advanced examples are reported to demonstrate the possibilities of the package. The examples are adapted from the website `http://texample.net`, which exhibits a gallery containing a large number of high quality illustrations and graphics made with tikz and pgf.

The LATEX code reported in Figure 10 produces the example of Figure 11, a nice block diagram obtained with tikz. The diagram is constructed with the aid of the tikz library positioning, which facilitates the relative positioning of the various nodes on the canvas. The code also demonstrates the definition of custom node styles.

Figure 12 shows the geometry of hydrogen and oxygen atoms in the water molecule. The example demonstrates the use of shading options to obtain a three-dimensional effect.

Figure 13 demonstrates the way a submatrix can be highlighted within a mathematical formula. In this example some advanced features of pgf are used in order to produce rectangle nodes that fit the desired areas. The drawing requires a double compilation.

7 LATEX-aware graphic software

The approach to graphic work production discussed in this section relies on available visual tools and is very different from the 'programmed graphics' approach presented in the previous section.

There are many 'LATEX-aware' computer programs, with sophisticated graphical user interfaces (GUI), not included in standard TEX distributions, which are capable of producing professional-quality graphics. The following is a list of the most popular ones:

- Xfig,[14] is one of the first visual tools of this type, an X Window drawing software available for Unix and Linux that saves graphics in its own format (`.fig` files), but exports to many other formats, including Encapsulated PostScript (EPS). An improved version named WinFig[15] is available for MS Windows. This software has been traditionally used in conjunction with the LATEX package psfrag that can remove labels and other text from `.eps` graphics and replace them with LATEX labels. Although this approach still works perfectly, it has become somewhat obsolete with respect to other recently introduced workflows.

- Ipe,[16] is a powerful vector graphics editor, with several snapping modes that make it especially suitable for a variety of technical illustrations. The application saves graphics in its own `.ipe` file format, but outputs PDF and EPS for inclusion in LATEX documents. Ipe uses LATEX to typeset text,

[13] `https://www.overleaf.com/read/mgskyfdpttzt`

[14] `http://xfig.org`

[15] `http://winfig.com`

[16] `http://ipe7.sourceforge.net`

```
%in preamble
\usepackage{relsize,calc,paralist,tikz}
\usetikzlibrary{calc,arrows,decorations.pathmorphing,backgrounds,fit,positioning,shapes.symbols,chains}

\definecolor{mydarkgreen}{rgb}{0.03,0.47,0.03} \definecolor{mydarkblue}{rgb}{0.07,0.08,0.4}
\definecolor{mylightblue}{rgb}{.8, .8, 1} \definecolor{mylightgray}{rgb}{0.95,0.95,0.95}
\definecolor{mydarkgray}{rgb}{0.35,0.35,0.35} \definecolor{myblue}{rgb}{.4,.4,1}

\tikzstyle{line} = [draw,>=latex’, shorten >=0pt, shorten <=0pt,line width=2pt]

% in document -------------------------------------------
\begin{tikzpicture}
   [node distance = 1cm, auto, font=\footnotesize,
   % STYLES
   every node/.style={node distance=3cm},
   % The comment style is used to describe the characteristics of each force
   comment/.style={
      rectangle, inner sep= 2pt, text width=5cm, node distance=0.25cm,
      font=\relsize{0}\sffamily
   },
   % The discipline style is used to draw the disciplines’ name
   discipline/.style={
      rectangle, draw, fill=black!10, inner sep=5pt, text width=4cm, text badly centered,
      minimum height=1.2cm, font=\relsize{0}\bfseries
   },
   % The topic style
   topic/.style={
      rectangle, draw, top color=white, bottom color=mylightblue,very thick,
      inner sep=5pt, text width=3.5cm, text badly centered,
      minimum height=1.7cm, font=\relsize{0}\bfseries
   },
   % the mycircled node type
   mycircled/.style={
      circle, draw, fill=black!10, inner sep=2pt,font=\relsize{0}\bfseries
   }
]% end of tikzpicture global options

% Draw forces
\node [discipline] (FD) {\relsize{1}Flight Dynamics};
\node [mycircled, left of=FD] (plus) {$\boldsymbol{+}$};

\node [discipline, left of=plus] (MechElasStru) {Mechanics of Elastic Structures};
\node [discipline, above of=MechElasStru,yshift=-1cm] (MechRigiBodi) {Mechanics of Rigid Bodies};
\node [discipline, above of=MechRigiBodi,yshift=-1cm] (Aero) {Aerodynamics};
\node [discipline, below of=MechElasStru,yshift=+1cm] (HumaPiloDyna) {Human Pilot Dynamics};
\node [discipline, below of=HumaPiloDyna,yshift=+1cm] (ApplMathMachComp) {Applied Mathematics Machine Computation};

\node [discipline, right of=FD,xshift=3cm] (VehiOper) {Vehicle Operation};
\node [discipline, above of=VehiOper,yshift=-1cm] (VehiDesi) {Vehicle Design};
\node [discipline, below of=VehiOper,yshift=+1cm] (PiloTrai) {Pilot Training};

\node [topic, below of=FD,xshift=-5.98cm, yshift=-3.8cm] (P) {Performance (trajectory, maneuverability)};
\node [topic, right of=P,xshift=1cm] (SC) {Stability \& Control (handling qualities, airloads)};
\node [topic, right of=SC,xshift=1cm] (AE) {Aeroelasticity (control, structural integrity)};
\node [topic, right of=AE,xshift=1cm] (NG) {Navigation and Guidance};

% Comments
\node [comment, above=0.25 of FD] (comment-FD) {
  \begin{compactitem}% needs paralist
    \item Flight Simulator mathematical model \item Aircraft representation
  \end{compactitem}
};

% Draw the links between nodes
\path[line,->] (plus) edge (FD);
\path[line,->] (MechElasStru) edge (plus);
\path[line,->] (MechRigiBodi.east) -- ++(0.3cm,0) -- (plus.120);
\path[line,->] (Aero) -| (plus);
\path[line,->] (HumaPiloDyna.east) -- ++(0.3cm,0) -- (plus.240);
\path[line,->] (ApplMathMachComp) -| (plus);

\path[line,->] (FD) -- (VehiOper);
\path[line,->] (FD.east) ++(1cm,0) |-  (VehiDesi);
\path[line,->] (FD.east) ++(1cm,0) |-  (PiloTrai);

\path[line,<-] (P.north) -- ++(0,0.6cm) -|  (FD);
\path[line,<-] (SC.north) -- ++(0,0.6cm) -|  (FD);
\path[line,<-] (AE.north) -- ++(0,0.6cm) -|  (FD);
\path[line,<-] (NG.north) -- ++(0,0.6cm) -|  (FD);
\end{tikzpicture}
```

Figure 10: Source code of diagram reported in Figure 11.

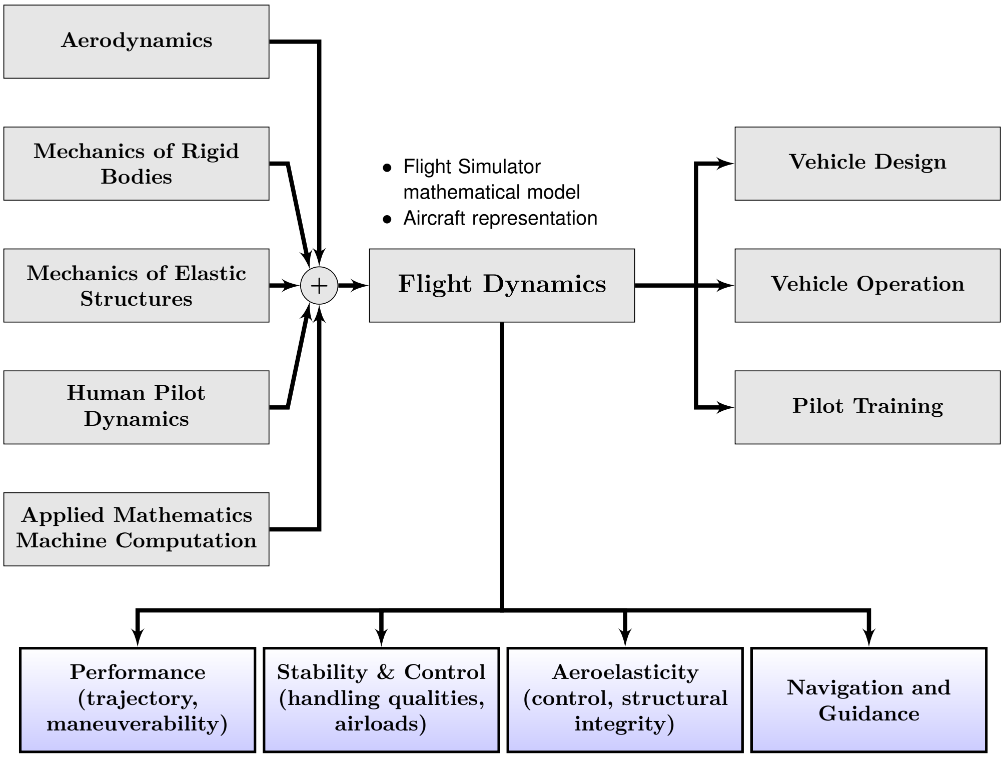

Aerodynamics
Mechanics of Rigid Bodies
Mechanics of Elastic Structures
Human Pilot Dynamics
Applied Mathematics Machine Computation
+
• Flight Simulator mathematical model
• Aircraft representation
Flight Dynamics
Vehicle Design
Vehicle Operation
Pilot Training
Performance (trajectory, maneuverability)
Stability & Control (handling qualities, airloads)
Aeroelasticity (control, structural integrity)
Navigation and Guidance

Figure 11: Block diagram of disciplines involved in Flight Dynamics. All graphic elements are placed on the canvas with intuitive tikz commands. See code in Figure 10.

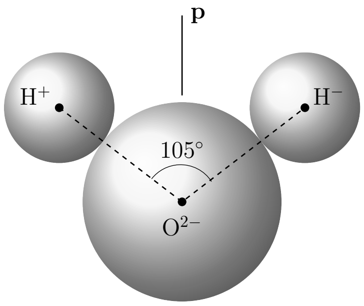
both simple labels and larger paragraphs. Supports layers and views, which make it possible to 'build' illustrations incrementally in a presentation.

- Asymptote,[17] is a vector graphics language and compiler. This software has been mentioned earlier in this article because code snippets in Asymptote language can be embed in LaTeX sources. Asymptote compiler can be used as a standalone tool as well (comes also with its own GUI) for generating both 2D and 3D figures. 3D figures can be included in a pdf file in the PRC (Product Representation Compact) format which allows them to be manipulated when viewed in Adobe Reader.
- LaTeXPiX,[18] is a Windows GUI capable of exporting pgf/LaTeX code.
- TPX,[19] is a Windows GUI similar to LaTeXPiX, but more flexible.
- Sweave,[20] is a tool that allows users to include R code directly into their LaTeX files. It does much more than just generate graphics, but it makes inclusion of R generated graphics into LaTeX document very easy.
- KtikZ/QtikZ,[21] is a pgf/tikz real-time open source compiler that runs on Linux and Windows. It can speed up the drawing effort while at the same time allowing to code directly in tikz language. It has a template option which allows to define user commands in an easy way as well as a menu with many common (and not so common) tikz constructs.
- LatexDraw,[22] is a very useful open source multiplatform GUI capable of generating pstricks code.
- Dia,[23] is a multiplatform open source GUI that supports both pgf/tikz and pstricks output.
- Sketch,[24] is a language and compiler that allows users to create vector drawings of 3D scenes. It generates pgf/tikz or pstricks code. A detailed presentation of this software can be found in De Marco (2007).
- Inkscape,[25] is an open source and well-supported vector graphics/SVG editor available for all major operating systems. Due to its popularity, and being by far the most powerful and important application among those mentioned here, we will discuss the LaTeX-related capabilities of this graphics software in the rest of this section.

7.1 Using Inkscape

Inkscape is an open source vector graphics editor using the W3C standard Scalable Vector Graphics (SVG) file format, with capabilities similar to

[17] http://asymptote.sourceforge.net
[18] http://latexpix.comyr.com/latexpix.htm
[19] http://tpx.sourceforge.net
[20] http://www.stat.uni-muenchen.de/~leisch/Sweave
[21] http://www.hackenberger.at/blog/ktikz-editor-for-the-tikz-language
[22] http://latexdraw.sourceforge.net
[23] http://live.gnome.org/Dia
[24] http://www.frontiernet.net/~eugene.ressler
[25] http://www.inkscape.org



(a) The diagram by Jimi Oke shows the geometry of hydrogen and oxygen atoms in the water molecule, and the position of the dipole (p).

```
\begin{tikzpicture}[>=latex,scale=1.3]
  \shade[ball color=gray!10!] (0,0)
    coordinate(Hp) circle (0.9);
  \shade[ball color=gray!10!] (2,-1.53)
    coordinate(O) circle (1.62);
  \shade[ball color=gray!10!] (4,0)
    coordinate(Hm) circle (0.9);
  \draw[thick,dashed] (0,0) -- (2,-1.53)
    -- (4,0) ;
  \draw[thick] (2,.2) -- (2,1.5) node[
    right]{$\mathbf{p}$};
  \draw (2.48,-1.2) arc (33:142:.6);
  \draw (2,-.95) node[above]{$105^{\circ}
    $};
  \draw (0,.2) node[left]{H$^+$};
  \draw (4,.2) node[right]{H$^-$};
  \draw (2,-1.63) node[below]{O$^{2-}$};
  \foreach \point in {O,Hp,Hm}
    \fill [black] (\point) circle (2pt);
\end{tikzpicture}
```

(b) The tikz code of the above drawing.

Figure 12: Example of graphics made with tikz.

commercial applications such as Adobe Illustrator, CorelDRAW, or Xara X.

Inkscape supports many advanced SVG features, moreover, developers took great care in designing a streamlined interface, that allows user to edit nodes, perform complex path operations, trace bitmaps, and much more in a very easy way. A well written documentation and many tutorials are available online. For a guide on this application the reader is referred to Bah (2011).

Inkscape provides a large API (Application Programming Interface) and a Python scripting capability. These features have encouraged the development of several third-party extension plugins. One of these plugins is TexText,[26] a particularly important extension for TeX users because it gives them the possibility to add and re-edit (multi-line) LaTeX/XeLaTeX/LuaLaTeX generated SVG elements to a drawing. It offers a multi-line editor, optionally with syntax highlighting.

SVG elements created with TexText are enriched with special additional information containing the LaTeX code used to generate all text and symbols prior to be traced into SVG vector graphics. The additional information allow users to re-edit the original LaTeX commands.

TexText is written in Python and uses either pdf2svg (or a combination of pstoedit and ghostscript) as converter for producing SVG code from the generated PDF (or PostScript). Detailed installation instructions for all major platforms are found on the project website.

Once TexText and its dependencies are correctly installed, a menu entry Extensions → Tex Text will appear in Inkscape. See Figure 15a. When this menu item is selected, a TexText dialog window appears to assist the user to input the desired LaTeX content.

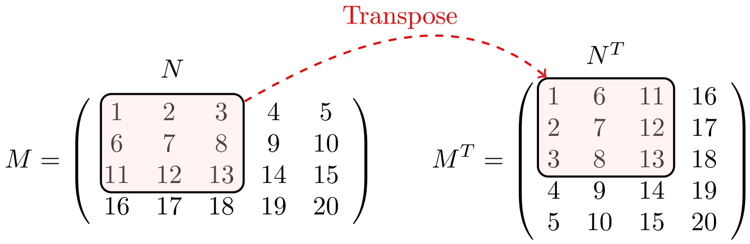
The TexText input dialog window is shown in Figure 15b. LaTeX code is entered into the edit box ❺. In the case PyGTK is installed, it will show line and column numbers. If PyGTKSourceView has been additionally installed, the edit box will also highlight the syntax with colors. The user can add any valid and also multi-line LaTeX code. The plugin provides additional settings which can be adjusted to the user's needs:

- The group box ❶ controls the TeX command to be used for compiling the code. Possible options are: pdflatex, xelatex, lualatex.
- The group box ❷ points to a custom preamble file that the user might need in order to have his LaTeX input compiled successfully.
- The group box ❸ regulates a scale factor to be applied to the final SVG element.
- The button ❹ becomes active only when the user re-edits the code of the enriched SVG element, and controls the alignment relative to the previous state of the same graphic object.
- Menu items ❼ control the default math environment in new nodes and the appearance of the editor.

The LaTeX code and the accompanying settings will be stored within the new SVG node in the Inkscape document. This allows the user to re-edit the 'LaTeX node' later by selecting it and running the TexText extension (which will then show the dialog containing the saved values).

The TexText dialog window provides also a preview button ❻ as well, which shortens the feedback cycle from entry to result considerably. Once the user is happy with the previewed LaTeX object the Save button can be clicked to get the resulting SVG object of Figure 16. The final traced result of the intermediate temporary PDF produced with the

[26] https://textext.github.io/textext



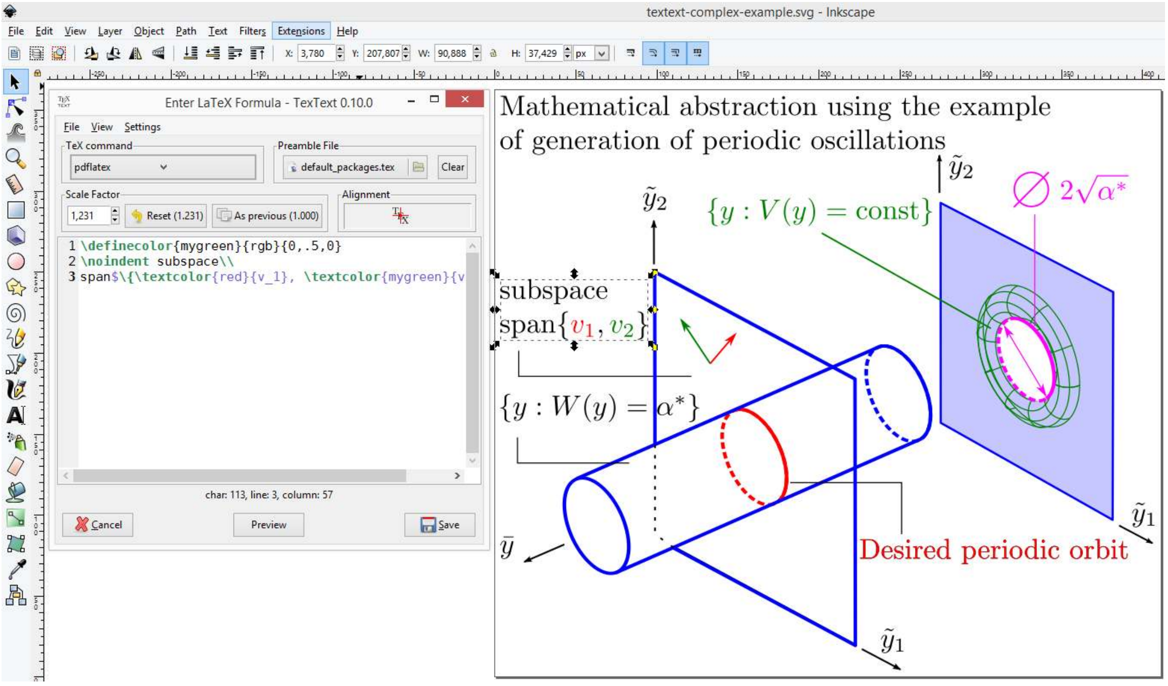
(a) An example by Stefan Kottwitz. A submatrix within a matrix is highlighted with tikz. The same is done in the transposed matrix. tikz and some advanced features of pgf are used in order to produce rectangle nodes that fit the desired areas. The drawing requires a double compilation.

```
% in preamble
\usepackage{tikz}
\usetikzlibrary{fit}
% ...
\tikzset{highlight/.style={rectangle
    ,rounded
    corners,fill=red!15,draw,fill
    opacity=0.3,thick,inner
    sep=0pt}}
\newcommand{\tikzmark}[2]{\tikz[
    overlay,remember
    picture,baseline=(#1.base)] \
    node (#1) {#2};}
\newcommand{\Highlight}[1][
    submatrix]{\tikz[overlay,
    remember
    picture]{
    \node[highlight,fit=(left.north
    west) (right.south east)] (#1)
    {};}}
```

```
\begin{document}
\[
  M = \left(\begin{array}{*5{c}}
    \tikzmark{left}{1} & 2 & 3 & 4 &
     5\\ 6 & 7 & 8 & 9 & 10 \\
    11 & 12 & \tikzmark{right}{13} &
     14 & 15 \\
    16 & 17 & 18 & 19 & 20 \end{
    array}\right)
  \Highlight[first]
  \qquad
  M^T = \left(\begin{array}{*5{c}}
    \tikzmark{left}{1} & 6 & 11 & 16
     \\ 2 & 7 & 12 & 17 \\
    3 & 8 & \tikzmark{right}{13} &
    18 \\ 4 & 9 & 14 & 19 \\
    5 & 10 & 15 & 20 \end{array}\
    right)
\]
\Highlight[second]

\tikz[overlay,remember picture] {
  \draw[->,thick,red,dashed] (first)
     to[out=30,in=140]
     node[above] {Transpose} (second
    );
  \node[above of=first] {$N$};
  \node[above of=second] {$N^T$};
}
\end{document}
```

(b) The tikz code of the above drawing.

Figure 13: Example of nice graphics made with tikz.

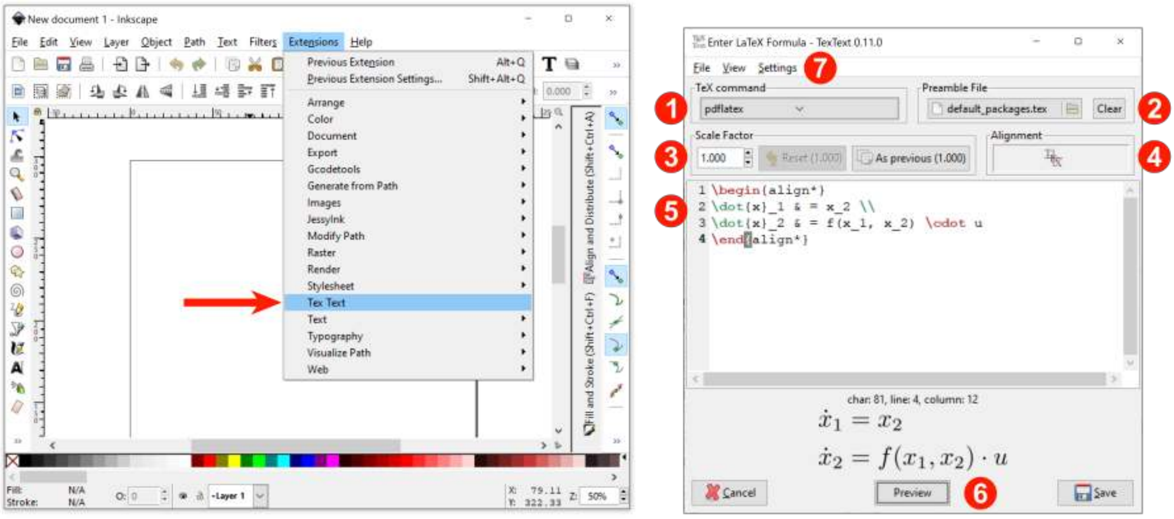

textext-complex-example.svg - Inkscape
File Edit View Layer Object Path Text Filters Extensions Help
Enter LaTeX Formula - TexText 0.10.0
File View Settings
Mathematical abstraction using the example of generation of periodic oscillations
subspace
span$\{v_1, v_2\}$
$\{y : W(y) = \alpha^*\}$
$\{y : V(y) = \text{const}\}$
$\tilde{y}_2$
$\tilde{y}_2$
$\varnothing\ 2\sqrt{\alpha^*}$
$\bar{y}$
$\tilde{y}_1$
$\tilde{y}_1$
Desired periodic orbit
Cancel
Preview
Save

TeX command: pdflatex
Preamble File: default_packages.tex
Scale Factor: 1,231 — Reset (1.231) — As previous (1.000)
Alignment

```
\definecolor{mygreen}{rgb}{0,.5,0}
\noindent subspace\\
span$\{\textcolor{red}{v_1}, \textcolor{mygreen}{v
```

char: 113, line: 3, column: 57

Figure 14: A screenshot of Inkscape with TexText extension in use.



TeX command: pdflatex
Preamble File: default_packages.tex
Scale Factor: 1.000 — Reset (1.000) — As previous (1.000)
Alignment

```
\begin{align*}
\dot{x}_1 & = x_2 \\
\dot{x}_2 & = f(x_1, x_2) \cdot u
\end{align*}
```

char: 81, line: 4, column: 12

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = f(x_1, x_2) \cdot u$$

(a) Selecting TexText from Inkscape Extensions menu.

(b) The TexText dialog window.

Figure 15: Using TexText extension plugin in Inkscape.

input LaTeX code is visible in Figure 17 where the highlighted nodes of the SVG element are shown.

TexText is a very powerful tool when coupled with the potential of Inkscape itself. Yet the user have to be aware of including the required packages in the preamble file if special commands are used in LaTeX code that rely on such packages. The preamble file can be chosen by the selector mentioned above. The default preamble file shipped with TexText is the following:

```
% default_packages.tex
\usepackage{amsmath,amsthm,amssymb,
    amsfonts}
\usepackage{color}
```

Basically, user's LaTeX code will be inserted into this template:

```
\documentclass{article}
% ===> preamble file content <===
% default:
%    \input{default_packages}
\pagestyle{empty}
\begin{document}
% ==> User's code <===
\end{document}
```

This will be typeset in a separate system thread, the PDF result will be converted to SVG and

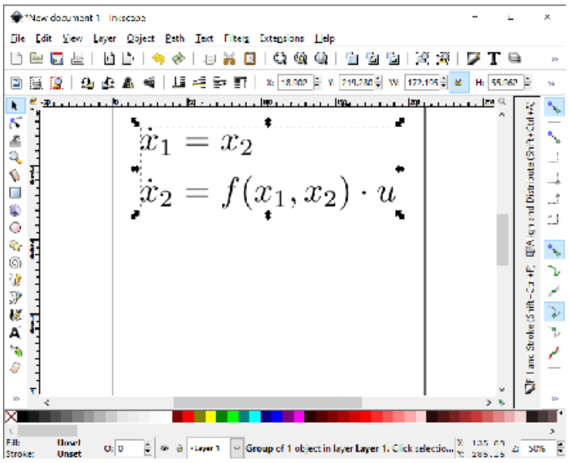

*New document 1 - Inkscape
File Edit View Layer Object Path Text Filters Extensions Help
$x_1 = x_2$
$\dot{x}_2 = f(x_1, x_2) \cdot u$
Fill: Unset
Stroke: Unset
Group of 1 object in layer Layer 1. Click selection...

Figure 16: SVG element resulting from input of Figure 15b.

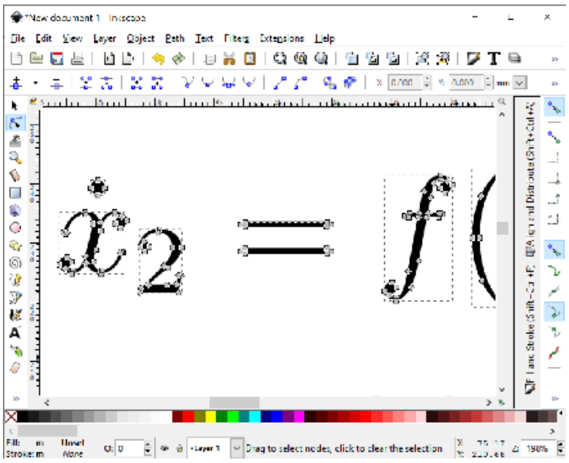

*New document 1 - Inkscape
File Edit View Layer Object Path Text Filters Extensions Help
$\dot{x}_2 = f($
Drag to select nodes, click to clear the selection

Figure 17: Nodes highlighted in the SVG element of Figure 16.

the vector object will be inserted into the current Inkscape document.

In conclusion of this section, a fairly elaborated illustration is displayed in Figure 18. The vector image has been produced using Inkscape with the TexText extension. The preamble file has been customized to load the package mt2pro and use the commercial font MathTime Professional 2.

8 Presenting data with plots

This section studies the presentation of data with “data plots” using LaTeX. Usually one shall use the word ‘graph’ instead of data plot. The main focus of this final part of the article is the package pgfplots, which creates astonishingly beautiful data plots in a consistent style with great ease.

The package pgfplots is built on top of pgf and is designed to draw graphs in a variety of formats, with a consistent, professional look and feel. The package also allows to import data stored in files in tabular format via the package pgfplotstable.[27]

As is usual with the pgf family, their manuals are impressive (Feuersänger, 2018).

8.1 The axis environment

The workhorse of the pgfplots package is an environment called axis, which may define one or several *plots* (graphs). Each plot is drawn with the command \addplot. When the graphs are drawn the environment also draws a 2- or 3-dimensional axis. The axis environment is used inside a tikzpicture environment, so one can also use tikz commands. The options of the axis environment specify the type of the plot, the width, the height, and so on.

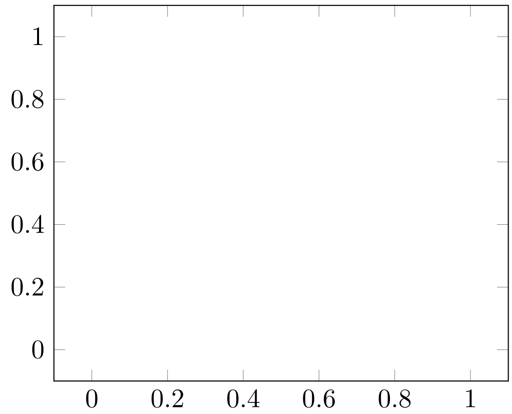
Typically, one or more plots are created in LaTeX following the template:

```
% in preamble
\usepackage{pgfplots}% loads tikz
...
\begin{tikzpicture}
  \begin{axis}[⟨graphic options⟩]
  ...
  ⟨pgfplots or tikz commands⟩
  ...
  \end{axis}
\end{tikzpicture}
```

The simplest possible graph with pgfplots is given by the code

```
\begin{tikzpicture}
  \begin{axis}
  \end{axis}
\end{tikzpicture}
```

that is, an empty axis environment, with default formatting options. The result is:



This can be customized, for example, changing the ranges of x- and y-axis, introducing a grid, and defining axis labels. This is done by passing the following self-explanatory options to the axis environment

```
\begin{axis}[
  xmin = -1, xmax = 1,
  ymin =  0, ymax = 2,
  grid = major,
```

[27] https://ctan.org/pkg/pgfplotstable

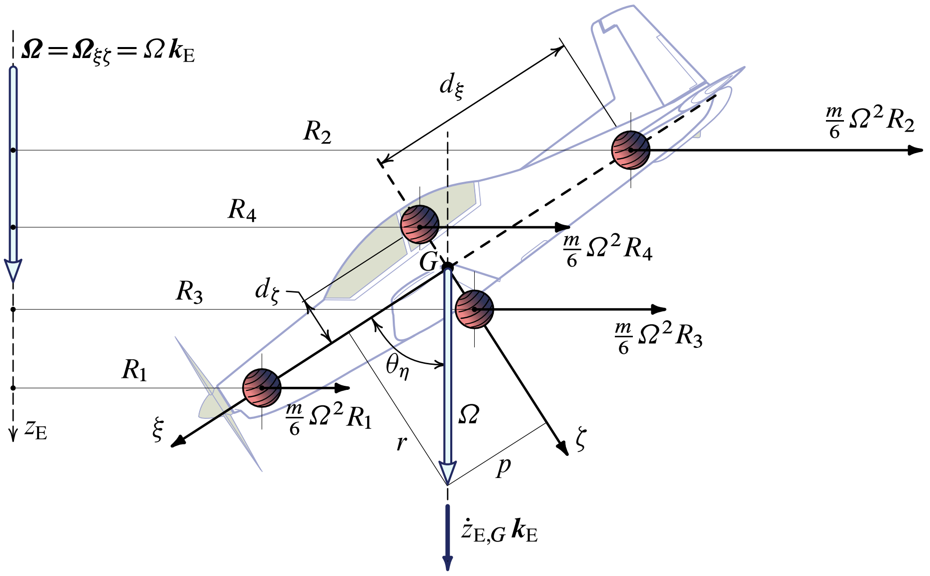
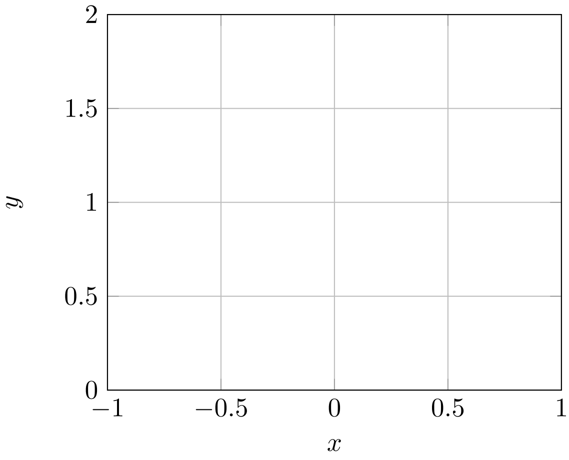

$\boldsymbol{\Omega} = \boldsymbol{\Omega}_{\xi\zeta} = \Omega\,\boldsymbol{k}_\mathrm{E}$
d_ξ
R_2
$\frac{m}{6}\,\Omega^2 R_2$
R_4
$\frac{m}{6}\,\Omega^2 R_4$
G
R_3
d_ζ
$\frac{m}{6}\,\Omega^2 R_3$
R_1
θ_η
z_E
ξ
$\frac{m}{6}\,\Omega^2 R_1$
Ω
ζ
r
p
$\dot{z}_{\mathrm{E},G}\,\boldsymbol{k}_\mathrm{E}$

Figure 18: A fairly elaborated illustration representing an aircraft in a spin manoeuvre. The image has been made using Inkscape with the TexText extension. The preamble file has been customized to load the package mt2pro and use the commercial font MathTime Professional 2.

```
  xlabel = $x$, ylabel = $y$
]
\end{axis}
```

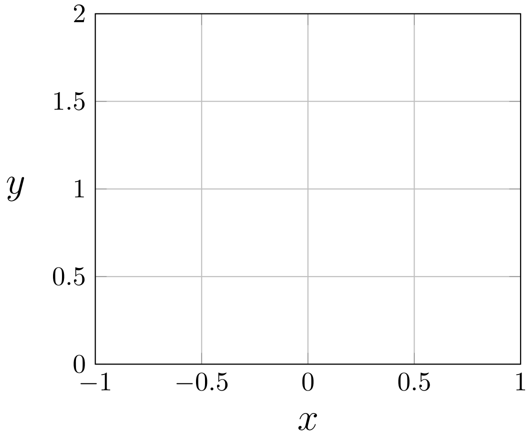
The customized axes now appear as follows:



The package pgfplot, as also pgf and tikz do, provides a way to change default settings at a global level. For plots this feature is given by the command \pgfplotsset. The following example enlarges the default font size in axis labels and rotates the y-axis label.

```
% in preamble
\usepackage{pgfplots,relsize}
...
% pgfplots styles
\pgfplotsset{
  every axis x label/.append style = {
    font = \relsize{2}
  },
  every axis y label/.append style = {
    font = \relsize{2},
    rotate = -90,
    xshift = 0.5em
  }
}
\begin{tikzpicture}
\begin{axis}[
  xmin = -1, xmax = 1,
  ymin =  0, ymax = 2,
  grid, xlabel = $x$, ylabel = $y$
]
\end{axis}
\end{tikzpicture}
```

The above code yields:

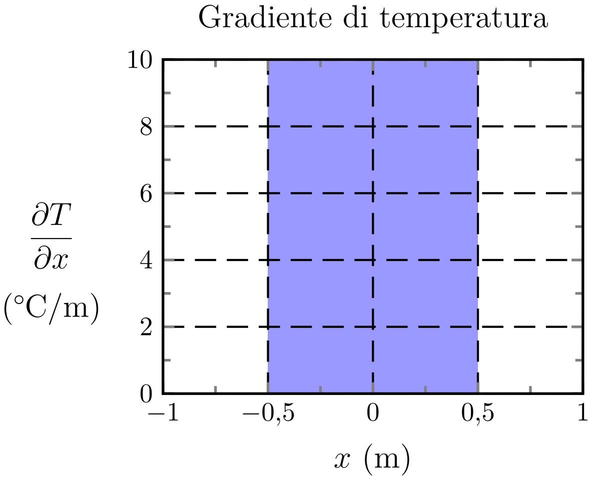


The macro \pgfplotset acts on predefined *styles*, as, for instance, in the last example on those labelled every axis x label as well as every axis y label. The usual approach is to change style parameters by *appending* customized settings to the defaults, such as font, rotate, xshift, and several others. The user provided settings overwrite the default ones. To learn more on style customizations available for pgf and pgfplots the reader might want to look at the remaining examples in this section and at the package documentation.

The following example demonstrates further customizations. The style of grid lines is changed and axis labels are formatted with the help of macro \si provided by the package siunitx.

```
% in preamble
\usepackage{pgfplots,relsize,siunitx}
...
% pgfplots styles
\pgfkeys{
 /pgf/number format/.cd,
 use comma
}
\pgfplotsset{
 every axis/.append style={
  font=\relsize{0},
  line width=1.0pt,
  tick style={line width=1.0pt}
  },
 every axis x label/.append style={
  font=\relsize{1},
  yshift=0pt,
  xshift=0em
 },
 every axis y label/.append style={
  font=\relsize{1},
  rotate=-90,
  xshift=-0.7em,
  yshift=-1.4em,
 },
 major grid style={
  line width = 0.8pt,
  black,
  dash pattern=on 8pt off 4pt
 },
 every axis title/.append style={
  font=\relsize{1}
 }
}
\begin{tikzpicture}
\begin{axis}[
 xmin=-1, xmax=1,
 ymin=0, ymax=10,
 xtick={-1,-0.5,...,1},
 ytick={0,2,...,10},
 minor x tick num = 1,
 minor y tick num = 1,
 grid=major,
 xlabel={$x$ (\si{\meter})},
 ylabel={
  \parbox{2cm}{%
   \centering
   $\dfrac{\partial T}{\partial x}$
   \\[0.7em]
   \centering
   (\si{\celsius/\meter})
  }
 },
 title=Gradiente di temperatura,
 axis on top=true
]
% the shaded rectangle
\fill[blue!40]
  (axis cs:-0.5,0) --
  (axis cs:0.5,0) --
  (axis cs:0.5,10) --
  (axis cs:-0.5,10) --
  cycle;

\end{axis}
\end{tikzpicture}
```

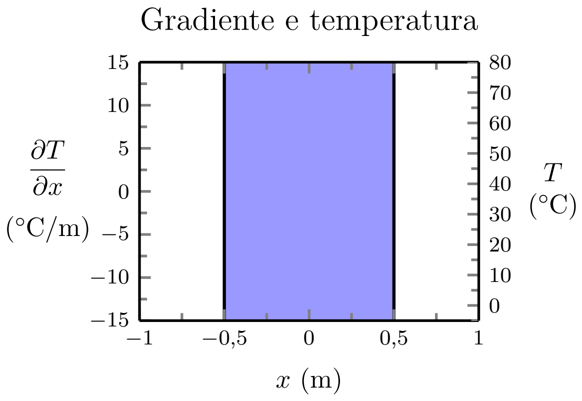
The macro pgfkeys, similar to pgfplotsset, is provided by pgf and is used to change the default decimal separator in numbers from '.' (dot) to ',' (comma). This example demonstrates also the use of tikz drawing commands inside the axis environment. The above code yields:



Next example evolves from the previous one. It demonstrates the use of a second y-axis on the right side of the plot bounding box. The second axis has different range and scaling with respect to the default one on the left side, and is used typically when multiple sets of data with values in different ranges have to be represented in the same plot.

```
\begin{tikzpicture}
% same initial settings
% of previous example
\begin{axis}[
 height=6cm,% <==
 % same initial settings
 % of previous example
]
% same drawing commands
% of previous example
% add plot data #1 here <==
\end{axis}
% second axis
\begin{axis}[
 height=6cm,% <==
 xmin=-1, xmax=1,
 axis x line=none,  % <==
 axis y line*=right,% <==
 ymin=-5, ymax=80,
 ytick={-10,0,...,80},
 minor y tick num = 1,
 ylabel={
   \parbox{2cm}{%
     \centering
     $T$
     \\[0.0em]
     \centering
     (\si{\celsius})
   }
 },
]
% add plot data #2 here
\end{axis}
\end{tikzpicture}
```

The new y-axis is obtained by superimposing a second reference frame on the first one. This is done by giving a second axis environment in the same tikzpicture. The second environment has an hidden x-axis and an `axis y line` set to `right`. All data whose y-values are conveniently represented with the new axis range should be provided in the second axis environment. The above code yields:



The code of a more elaborated multiple axis example is shown in Figure 19. The product is that of Figure 20 showing three y-axes conveniently positioned on the left- and right-hand sides of the main area of the plot. Various `tikz` drawing commands are used in this case to annotate and decorate the graph for a refined visual result.

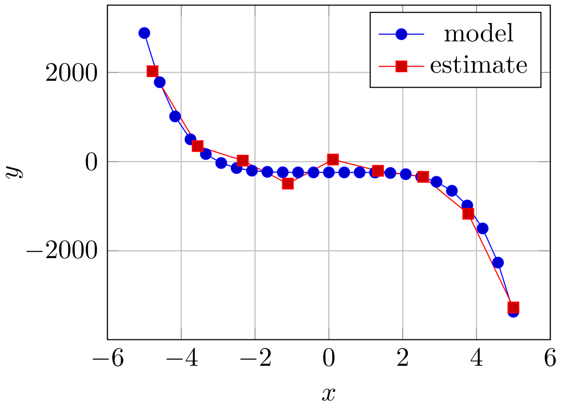
8.2 The macro \addplot

The command `\addplot` is used within an axis environment to define the lines in a graph. The command accepts a number of options and an argument that specifies the set of data to be represented on canvas.

The following example contains two line graphs with two different markers and a legend. No options are passed to the two `\addplot` commands to customize their behaviour. Yet, two different types of data sources are chosen: the first is a mathematical function, $f(x) = -x^5 - 242$; the second is a discrete set of (x, y)-coordinates.

```
\begin{tikzpicture}
\begin{axis}[
  grid=major,
  xlabel={$x$}, ylabel={$y$},
  y tick label style={
    /pgf/number format/.cd,
    set thousands separator={},
  /tikz/.cd}
  ]
  % a function of x
  \addplot {-x^5 - 242};
  \addlegendentry{model}
  % a discrete set of coordinates
  \addplot coordinates {
    (-4.77778, 2027.60977)
    (-3.55556,  347.84069)
    (-2.33333,   22.58953)
    (-1.11111, -493.50066)
    (0.11111,    46.66082)
    (1.33333,  -205.56286)
    (2.55556,  -341.40638)
    (3.77778, -1169.24780)
    (5.00000, -3269.56775)
  };
  \addlegendentry{estimate}
\end{axis}
\end{tikzpicture}
```

The above code yields:



The mathematical function is defined intuitively at high-level as `-x^5 - 242` and is parsed by the powerful low-level `\pgfmathparse` feature of pgf. By default, the function is evaluated at 25 points equally spaced between two automatically calculated x-axis limits — in this example $[-5, 5]$. Data points are connected with a blue solid line and marked by default with dots of the same color. The second set of data is manually given with a `coordinates`

```
%in preamble
\usepackage{pgfplots}
\usetikzlibrary{calc,arrows,decorations.pathmorphing,
     backgrounds,fit,positioning,shapes.symbols,shapes.
     geometric,shapes.misc,chains}
% ...
\begin{tikzpicture}
\pgfplotsset{compat=1.3}
\pgfkeys{
   /pgf/number format/.cd,
       set decimal separator={,{\!}},
      set thousands separator={}}
\pgfplotsset{
   every axis/.append style={
      font=\relsize{2},
      line width=1.0pt,
      tick style={line width=1.0pt}},
   every axis x label/.append style={
      font=\relsize{4},
      yshift=0pt,
      xshift=0em},
   every axis y label/.append style={
      font=\relsize{3},
      rotate=-90,
      xshift= 0.8em,
      yshift=-1.4em},
   major grid style={
      line width = 0.8pt,
      black,
      dash pattern=on 8pt off 4pt},
   every axis title/.append style={
      font=\relsize{3}},
   no markers
}
% the left y-axis #1
\begin{axis}[
   clip=false,
   scale only axis,
   width=2cm, xshift=-0.4cm,
   xmin=-1, xmax=1,
   hide x axis,
   axis y line*=left,
   ymin=-15, ymax=15,
   ytick={-15,-10,...,15},
   minor y tick num = 1]
\node [above, yshift=6pt] at (rel axis cs:0,1)
   {$\dfrac{\partial T}{\partial x}$ (\si{\celsius/\meter
     })};
\end{axis}
% the unique x-axis
\begin{axis}[
   scale only axis,
   height=2cm, yshift=-0.4cm,
   xmin=-1, xmax=1,
   xtick={-1,-0.5,...,1},
   minor x tick num = 1,
   xlabel={$x$ (\si{\meter})},
   axis x line*=bottom,
   hide y axis,
   ymin=-15, ymax=15,
]
\end{axis}
```

```
% the curve #1
\begin{axis}[
   scale only axis,
   xmin=-1, xmax=1,
   hide x axis,
   ymin=-15, ymax=15,
   hide y axis,
   title=\parbox{8cm}{\centering Trasmissione del calore
     attraverso una parete},
]
\fill[blue!40]
    decorate [decoration={random steps,segment length=2mm
     }] { [very thick] (axis cs:-0.5,-14.8) -- (axis cs
     :0.5,-14.8) } -- (axis cs:0.5,14.8)
    decorate [decoration={random steps,segment length=2mm
     }] { [very thick] -- (axis cs:-0.5,14.8)}
    -- cycle;
\draw[very thick] (axis cs:-0.5,-15) -- (axis cs:-0.5,15)
     ;
\draw[very thick] (axis cs:0.5,-15) -- (axis cs:0.5,15);
\node [rounded rectangle, minimum size=6mm, very thick,
     draw=black!50, top color=white, bottom color=black
     !20, font=\ttfamily] at (rel axis cs:0.125,0.94)
     {1};
\node [rounded rectangle, minimum size=6mm, very thick,
     draw=black!50, top color=white, bottom color=black
     !20, font=\ttfamily] at (rel axis cs:0.50,0.94) {2};
\node [rounded rectangle, minimum size=6mm, very thick,
     draw=black!50, top color=white, bottom color=black
     !20, font=\ttfamily] at (rel axis cs:0.875,0.94)
     {3};
% \addplot of temperature gradients here
\end{axis}
\pgfplotsset{
   every axis y label/.append style={
      xshift= -2.4em
   }
}
% the right y-axis 1
\begin{axis}[clip=false, scale only axis,
   xshift=0.4cm, xmin=-1, xmax=1, hide x axis,
   axis y line*=right,
   ymin=-5,ymax=80, ytick={-10,0,...,80},
   minor y tick num = 1,
]
\node [above, yshift=6pt] at (rel axis cs:1,1)
   {$T$ (\si{\celsius})};
% \addplot of temperatures here
\end{axis}
\pgfplotsset{every axis y label/.append style={xshift
     =-1.3em}}
% the right y-axis 2
\begin{axis}[clip=false, scale only axis,
   xshift=2.4cm,
   xmin=-1, xmax=1,
   axis x line=none, hide x axis,
   axis y line*=right,
   ymin=-5, ymax=500,
   ytick={0,50,...,500},
   minor y tick num = 1,
]
\node [above, yshift=6pt, xshift=16pt] at (rel axis cs
     :1,1) {$q$ (\si{\kcal/\meter^2})};
% \addplot of heat fluxes here
\end{axis}
\end{tikzpicture}
```

Figure 19: Source code of diagram reported in Figure 20.

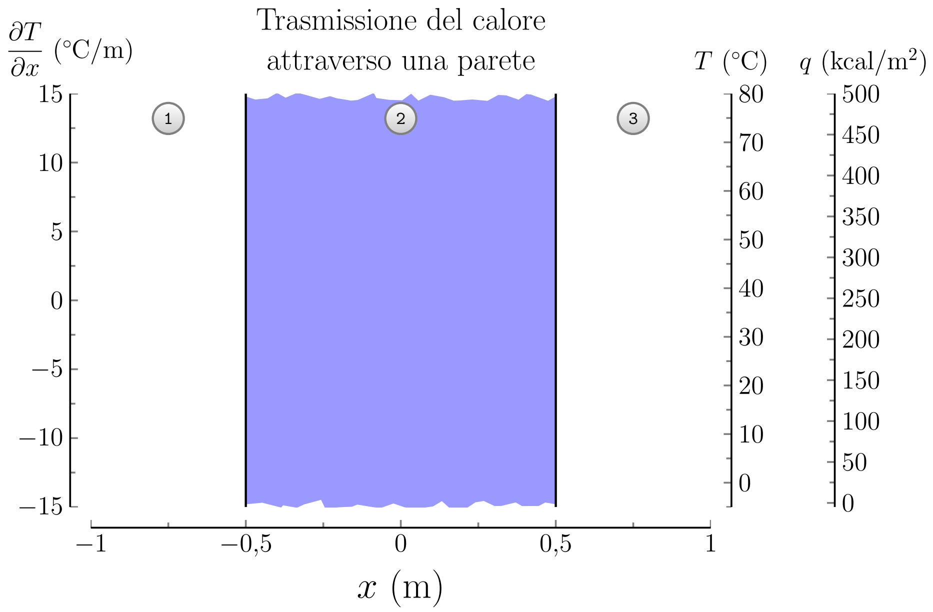


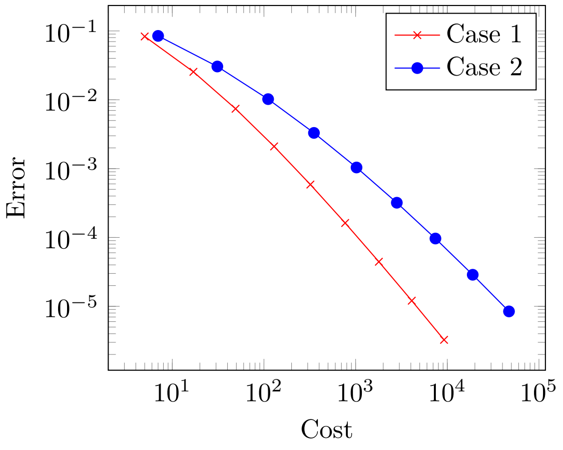
Figure 20: Example of multiple y-axes obtained with pgfplots. See source code in Figure 10.

directive to `\addplot` as a sequence of couples (⟨x⟩,⟨y⟩). This second set of data points are connected with a red solid line and marked by default with boxes filled with the same color.

The following example demonstrates the use of logarithmic scales for the axes within the environment loglogaxis. The markers of the line graphs are controlled by specific settings passed as options to the `\addplot` commands.

```
% in preamble
\usepackage{filecontents}
\begin{filecontents*}{data1.txt}
Level    Cost        Error
1          7 8.47178381e-02
2         31 3.04409349e-02
3        111 1.02214539e-02
4        351 3.30346265e-03
5       1023 1.03886535e-03
6       2815 3.19646457e-04
7       7423 9.65789766e-05
8      18943 2.87339125e-05
9      47103 8.43749881e-06
\end{filecontents*}
% ...
\begin{tikzpicture}
\begin{loglogaxis}[
  xlabel=Cost, ylabel=Error]
\addplot[color=red,mark=x] coordinates {
  (5,    8.31160034e-02)
  (17,   2.54685628e-02)
  (49,   7.40715288e-03)
  (129,  2.10192154e-03)
  (321,  5.87352989e-04)
  (769,  1.62269942e-04)
  (1793, 4.44248889e-05)
  (4097, 1.20714122e-05)
  (9217, 3.26101452e-06)
};
\addplot[color=blue,mark=*]
  table[x=Cost,y=Error]
    {data1.txt};
\legend{Case 1,Case 2}
\end{loglogaxis}
\end{tikzpicture}
```

The second line graph is constructed by reading coordinates from a conveniently formatted text file, `data1.txt`. The file contains three columns of data and a first row that provides the labels for each column. Options `x=` and `y=` in the second `\addplot` command select the desired x- and y-coordinate sets according to the column names. The above code yields:



The example below demonstrates the use of a logarithmic scale for the y-axis only.

```
\begin{tikzpicture}
\begin{semilogyaxis}[
\begin{tikzpicture}
\tikzset{
  every pin/.style={
    fill=yellow!50!white,
    rectangle, rounded corners=3pt,
    font=\tiny},
  small dot/.style={
    fill=black, circle,
    scale=0.3}
}
\begin{axis}[
  clip=false,
  title=How \texttt{axis description cs} works
]
\addplot {x^3};

% annotations
\node[small dot,
  pin={[pin distance=2cm]20:{$(0,0)$}}]
  at (axis description cs:0,0) {};
\node[small dot,
  pin=-30:{$(1,1)$}]
  at (axis description cs:1,1) {};
\node[small dot,
  pin=-90:{$(1.03,0.5)$}]
  at (axis description cs:1.03,0.5) {};
\node[small dot,
  pin=125:{$(0.5,0.5)$}]
  at (axis description cs:0.5,0.5) {};
\end{axis}
\end{tikzpicture}
```

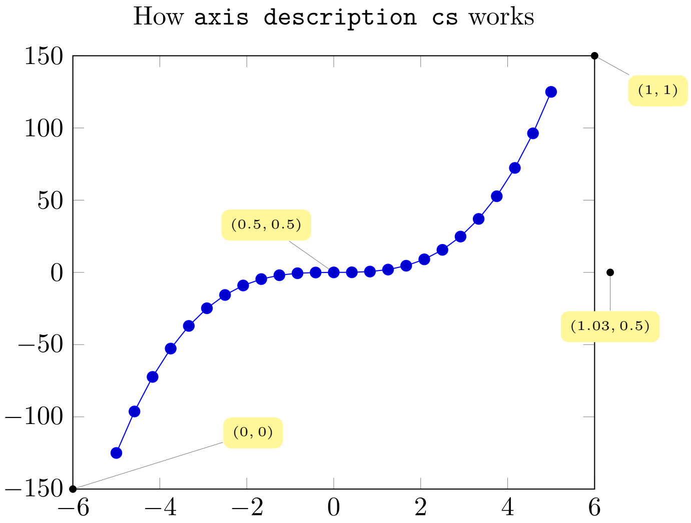
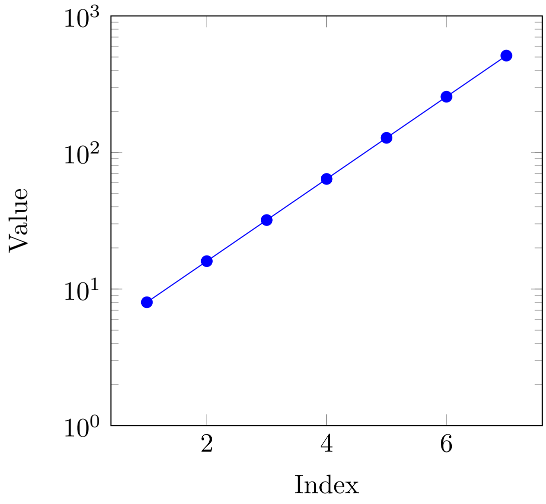


Figure 21: An example showing how, with the special coordinates system named `axis description`, in pgfplots it is possible to define points relative to the bounding box of a plot.

```
  ymin=1, ymax=1000,
  xlabel=Index,ylabel=Value]
  \addplot[color=blue,mark=*]
    coordinates {
      (1,8) (2,16) (3,32)
      (4,64) (5,128) (6,256)
      (7,512)
  };
  \end{semilogyaxis}
\end{tikzpicture}
```

The above code yields:



The package pgfplots defines special coordinate systems (`cs`) that make it easy to add annotations to the plots. One of these reference systems is named `axis description`. It has its point of coordinates $(0,0)$ in the bottom left corner of the plot bounding box, and its point $(1,1)$ at the top right corner of the frame. A demonstration of this coordinate system is shown in Figure 21. The annotations of the plot are made by defining a style/shape named `small dot`, and using the `pin` option of the tikz `\node` operation.

A fairly elaborated example of line graph annotation is provided by Figure 22. Thanks to the tikz library `intersection`, the pgf macro named `linelabel` is defined in such a way that it can be used as an option to `\addplot`. The option receives three arguments: the first is the normalized ascissa (in range $[0,1]$) of the point along the path where the annotation is pinned (0 for the leftmost point, 1 for the rightmost); the second argument specifies the angle and the length of the pin line; the third argument is the content of the annotation (a formula or even a multi-line text). This customization, too, is possible thanks to the `node` and `pin` features in tikz.

8.3 The macro \addplot3

The command `\addplot3` within an axis environment creates a three-dimensional plot. According to the option and arguments, it can display a line graph or a shaded surface.

The following example plots a helical curve:

```
\begin{tikzpicture}
\pgfplotsset{
```

```
\begin{tikzpicture}
% needs tikzlibrary: intersections
\pgfkeys{/pgfplots/linelabel/.style args
   ={#1:#2:#3}{
 name path global=labelpath,
 execute at end plot={
  \path [name path global =
   labelpositionline] (rel axis cs:#1,0)
    -- (rel axis cs:#1,1);
  \draw [help lines, text=black,
   inner sep=0pt,
   name intersections={
    of=labelpath and labelpositionline}]
    (intersection-1) -- +(#2)
    node [label={#3}] {};
  }
}}
\pgfplotsset{%
 every axis legend/.append style={
 cells={anchor=west},%
 fill=gray!10,
 font=\relsize{1},
 at={(0.97,0.03)},
 anchor=south east,thin,draw=none},
 every axis title/.append style={font=\
   relsize{1}},
 every axis/.append style={font=\relsize{0}},
 every axis x label/.append style={
  font=\relsize{1},
  yshift=0pt,xshift=0em},
 every axis y label/.append style={
  font=\relsize{2},
  rotate=-90},
 every axis/.append style={
  thick,
  tick style={thick}}
}
\tikzstyle{every pin}=[fill=white,draw=none,
   font=\relsize{2}]
\begin{axis}[xlabel={$x$}]
 \addplot [thick,
  linelabel=0.8:{135:1.75cm}:
   {[black]above left:$x^2$}] {x^2};
 \addplot [thick,
  blue,
  densely dashed,
  linelabel=0.85:{135:0.50cm}:
   $\frac{3}{2}x^2$] {1.5*x^2};
 \addplot [thick,
  red,
  dash pattern=on 5pt off 2pt,
  linelabel=0.7:{135:1.25cm}:
   $\frac{1}{10}x^3$] {0.1*x^3};
 \addplot [thick,
  dash pattern=on 1.2pt off 2pt on 5pt off 2pt,
  linelabel=0.80:{-135:0.75cm}:{left:
   \makebox[0pt][r]{$\left.
   \begin{array}{rl}
    \frac{1}{2}x^2 &\text{if }x\le 0\\[6pt]
    -\frac{1}{5}x^3 &\text{if }x> 0
   \end{array}
   \right\}$}
   }
  ]
  {(x<0)*0.5*x^2 + (x>0)*(-0.20*x^3)};
\end{axis}
\end{tikzpicture}
```

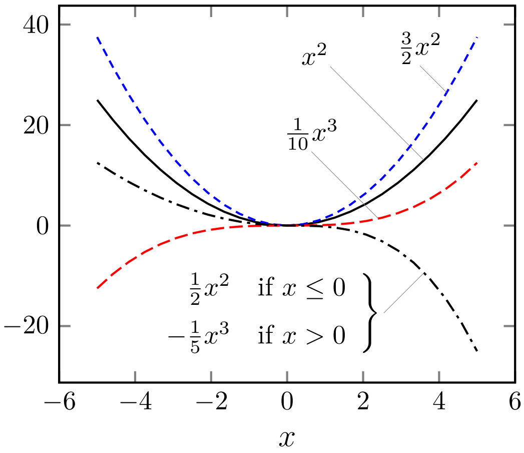


Figure 22: An example of line graph annotations obtained introducing a customized option `linelabel` to the command `\addplot`. The option works as a macro and is based on the `tikz` extension library `intersections`.

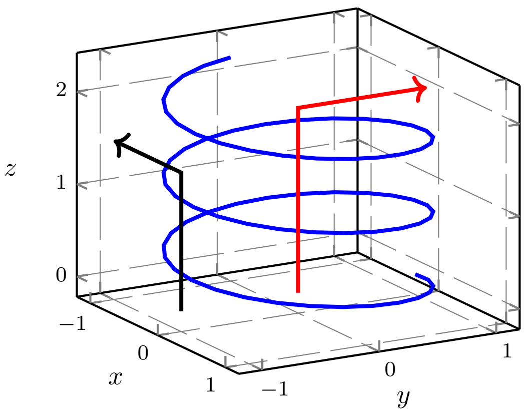
```
 every axis/.append style={
  font=\relsize{-1},
  line width=0.8pt,
  tick style={line width=0.8pt}
 },
 major grid style={
  line width = 0.4pt,
  gray,
  dash pattern=on 16pt off 4pt
 }
}
\begin{axis}[
  view={60}{20},% <== view point
  xmin=-1.2, xmax=1.2,
  ymin=-1.2, ymax=1.2,
  grid = major,
  xlabel=$x$, ylabel=$y$, zlabel=$z$,
  every axis x label/.style={
   at={(rel axis cs:0.5,-0.15,-0.15)},
   font=\relsize{0}},
  every axis y label/.style={
   at={(rel axis cs:1.15,0.5,-0.15)},
   font=\relsize{0}},
  every axis z label/.style={
   at={(rel axis cs:-0.15,-0.15,0.5)},
   font=\relsize{0}},
  variable=\t]
% the helix
\addplot3+[
 domain=0:5.5*pi,
 samples=70,
 samples y=0,
 no marks,
 line width=1.5pt]
 ( {sin(deg(t))},  % <== x(t)
  {cos(deg(t))},  % <== y(t)
  {2*t/(5*pi)} ); % <== z(t)
% a line in 3d
\addplot3 +[->,no marks,line width=1.5pt]
  coordinates {
    (0,0,0)
    (0,0,2)
    (0,1.1,2)};
% a line in 3d with a tikz command
\draw[->,line width=1.5pt]
  (axis cs:0,-1,0)
  -- (axis cs:0,-1,1.5)
  -- (axis cs:-1,-1,1.5);
\end{axis}
\end{tikzpicture}
```

The helix is defined as

$$x(t) = \sin t\,,\;\; y(t) = \cos t\,,\;\; z(t) = \frac{2}{5\pi}t$$

with $0 \leq t \leq \frac{11}{2}\pi$ (see domain option). The line graph is made by connecting 70 three-dimensional data points (see samples option). The mathematical expression of the curve is passed to \addplot3 as a triplet of functions of t after having declared variable=\t as an option of the axis environment. Two more lines are represented in the diagram. One is made with the \addplot3 command itself (hence, by default is red) by connecting three points: $(0, 0, 0)$, $(0, 0, 2)$, and $(0, 1.1, 2)$. The other line is made using the tikz command \draw by connecting three points: $(0, -1, 0)$, $(0, -1, 1.5)$, and $(-1, -1, 1.5)$.

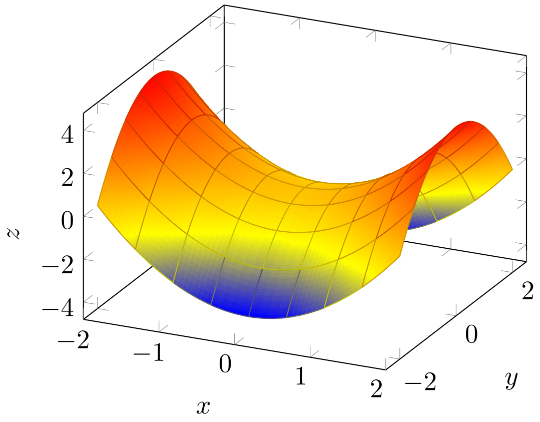
The above code yields:



The following final example creates the plot of a three-dimensional shaded surface. The surface is constructed by evaluating the function $f(x, y) = x^2 - y^2$ in a set of points on the xy-plane. The shading algorithm is provided by the pgfplots extension library patchplots.

```
\begin{tikzpicture}
\begin{axis}[width=0.98\linewidth,
  ymin=-2.5, ymax=2.5,
  xlabel={$x$}, ylabel={$y$}, zlabel={$z$
    }
]
\addplot3[
  % needs pgfplotslibrary: patchplots
  patch, patch refines=3,
  shader=faceted interp,
  patch type=biquadratic]
  table[z expr=x^2-y^2] {
     x  y
    -2 -2
     2 -2
     2  2
    -2  2
     0 -2
     2  0
     0  2
    -2  0
     0  0};
\end{axis}
\end{tikzpicture}
```

The above code yields:



8.4 What else?

In this final part we really have only scratched the surface of what can be done with pgfplots. A more in-depth presentation of both basic and advanced features of the package can be found in De Marco and Giacomelli (2011). Examples of quality manuscripts including several fine tuned technical illustrations, diagrams and scientific plots can be found on the author's page of his course on Flight Dynamics and Simulation at the University of Naples Federico II.[28]

Scientific writers nowadays can rely on several different applications and data visualization technologies for producing their own two- and three-dimensional graphs. These include, to name a few: Gnuplot, the Python libraries Matplotlib, Seaborn, ggplot, Bokeh, and Plotly, the R libraries ggplot2 and Lattice, the Javascript libraries D3 and Plotly.js, the numerical computing environments Matlab and Mathematica, and the highly specialized software Tecplot. Some of these tools provide their users with the possibility to export their plots as tikz or pgfplots code, e. g. the Gnuplot lua tikz terminal[29] or the Matlab script matlab2tikz.[30]

[28] http://wpage.unina.it/agodemar/DSV-DQV/#materiale-didattico

[29] http://gnuplot.info/documentation.html

[30] https://github.com/matlab2tikz/matlab2tikz

The strength of pgfplots combined with tikz is the excellent typographical quality of their graphical outputs. Apart from those given in this article and the pgfplots online gallery,[31] the reader can find several online examples of publication quality graphs.[32] Being pgfplots able to parse the definition of mathematical functions as well as to import numerical data produced with third-party software, a production work flow based on this package is probably the way to go for the majority of LATEX users. Moreover, this approach promotes the automation of production processes — from the collection of data, to their import in pgfplots, to the drawing and typesetting of the final image (in raster or vector format), up to the inclusion of the image in the master document. Experience confirms that this approach brings about a tremendous speed up of graphics production.

In cases where a certain third-party technology must be used to produce graphic works, still these can be successively annotated with LATEX content. The suggested approach is to import them in Inkscape and add LATEX objects with the TexText plugin. Examples of quality graphics matching the style of websites with a lot of LATEX content are given by the flight simulation software library JSBSim documentation project,[33] and by the online teaching material of the Flight Mechanics course for the Italian Air Force Academy student pilots.[34]

9 Conclusion

This paper describes the most common scenarios encountered by LATEX users when they face the problem of producing quality graphics to include in their documents. In cases of diagrams, pictures and more or less complicated illustrations the two approaches based on package tikz and on the Inkscape graphics vector software have been presented. The last part of the article introduces the package pgfplots for making scientific plots.

The paper is example driven and aims at stimulating readers' creativity, providing them as well with several online references.

References

Bah, Tavmjong (2011). *Inkscape. Guide to a Vector Drawing Program.* Prentice-Hall, Upper Saddle River, NJ, USA, 4th edition.

Beccari, Claudio (2011). «The unknown picture environment». *ArsTEXnica*, (11), pp. 57–64. http://www.guitex.org/home/it/numero-11.

De Marco, Agostino (2007). «Illustrazioni tridimensionali con Sketch/LATEX/PSTricks/TikZ nella didattica della Dinamica del Volo». *ArsTEXnica*, (4), pp. 51–68. http://www.guitex.org/home/numero-4.

— (2009). «Produrre grafica vettoriale di alta qualità programmando asymptote». *ArsTEXnica*, (8), pp. 25–39. http://www.guitex.org/home/numero-8.

De Marco, Agostino and Roberto Giacomelli (2011). «Creare grafici con pgfplots». *ArsTEXnica*, (12), pp. 12–38. http://www.guitex.org/home/it/numero-12.

Feuersänger, Christian (2018). «Manual for package pgfplots». https://ctan.org/pkg/pgfplots.

Goosens, Michel, Frank Mittelbach, Sebastian Rahtz, Denis Roegel and Herbert Voß (2007). *The LATEX Graphics Companion.* Addison-Wesley Publishing Company, Reading, Mass.

Harris, Robert L. (1996). *Information Graphics. A Comprehensive Illustrated Reference.* Management Graphics, Atlanta, GA, USA.

Lamport, Leslie (1994). *LATEX, a document preparation system.* Addison-Wesley, Reading, MA, 2nd edition.

Tantau, Till (2016). «The pgf package». http://ctan.org/pkg/pgf.

van Dongen, Marc (2012). *LATEX and Friends.* Springer-Verlag, Berlin, Heidelberg.

Voß, Herbert (2011). *PSTricks – Graphics and PostScript for TEX and LATEX.* UIT – Cambridge, Cambridge, UK, 1st edition.

▷ Agostino De Marco
Università degli Studi di Napoli
Federico II
agostino dot demarco at unina
dot it

[31] http://pgfplots.sourceforge.net/gallery.html

[32] From the author, see also these sample projects on Overleaf: https://www.overleaf.com/read/mgskyfdpttzt , https://www.overleaf.com/read/kqkvsrfjxnmz , https://www.overleaf.com/read/rcbqhpqqhccn .

[33] https://jsbsim-team.github.io/jsbsim-reference-manual

[34] https://agodemar.github.io/FlightMechanics4Pilots

Presentations with Beamer

Grazia Messineo, Salvatore Vassallo

Abstract

In this article we briefly introduce the LaTeX class beamer for presentation. We give some tips to build an effective presentation and we describe the main features of the class.

Sommario

In questo articolo si dà una breve introduzione alla classe LaTeX per presentazioni beamer. Vengono forniti alcuni suggerimenti per realizzare una presentazione efficace e descritte le principali funzioni della classe.

1 Introduction

For years, people thought that Microsoft PowerPoint® was the only tool to make presentations. More recently, Libre Office Impress® or Keynote®(for Mac) have been used for the same purpose.

When a presentation contains a great amount of mathematics and/or you want to keep its quality high you can use LaTeX packages to build it.

Many LaTeX classes have been developed over the years to write presentations. This paper focuses on the Beamer class, while a brief review of other classes will be made in paragraph 4.

2 Tips for a good presentation

Building a good presentation is not an easy task. Most presentations are boring, because they are too complicated, contain too many data, their organization is poor, the needs of the audience are not taken into account and perhaps the speaker reads it word by word. The result is an audience who gets lost after a few minutes.

In Tantau *et al.* (2015) there is a good tutorial for building presentations ("Euclid's presentation") with some guidelines to make them effective.

When building a presentation, the author should:

- *know the room*: it is a good idea to visit it before the presentation and be aware of the technical equipment available;
- *know the audience*: the author should know how it is composed, what they already know of the topic, which are their interests and what background information they need;
- *be aware of the time constraints.*

The presentation should state clearly at the beginning the purpose of the talk. The main body should contain intermediate conclusions (if possible) and then the final conclusion of the talk.

It should be clear and concise and focus on a limited number of concepts.

Each slide should focus on one concept and be at most 12 lines long. It should be shown and explain in one minute. The author should also schedule a question and answer moment for discussion.

Use of animations and special effects should be limited. Background should be chosen carefully, as well as colours.

A good use of images improves the quality of the presentation.

3 Presentations in Beamer

The Beamer class is an excellent tool for creating presentations, both for didactic and scientific purposes.

It was created in 2003 by Till Tantau for his PhD defense presentation and immediately published on CTAN. In 2010, maintenance was handed over to Joseph Wright and Vedran Miletić, who are still mantaining it. For the complete documentation of the class, please refer to Tantau *et al.* (2015).

The class gives its best for creating presentations to be displayed using (as the class name suggests) a beamer, but it can also be used to create transparency slides. It provides a great number of video "effects"[1] such as transitions, boxes and other animations.

Beamer output is a pdf file, so it is available on all platforms with a pdf files viewer[2].

Pdf files produced with the Beamer class are interactive, so each slide can also contain buttons to navigate into the presentation and it is possible to have a bar (side, bottom or top bar) containing an index, which is always visible and have hyperlink useful to move from a part to another of the presentation.

Beamer is a LaTeX class, so a presentation created with it has the same structure of all LaTeX documents: a preamble with all other packages and users' macros, a body divided into sections and subsections. Slides (Beamer *frames*) are inserted in

[1] The author and the mantainers, in the class manual, suggest not to exceed with video "effects" and colorful backgrounds, as they distract the audience.

[2] Not all pdf viewers can display correctly Beamer effects, such as transparencies.

each part of the body by using the corresponding environment.

Beamer can be used with pdfLATEX, LATEX + dvips, XƎLATEX and LuaLATEX.

As pointed out in Voss (2012), the main features of the class are:

- possibility to create printed, projected and noted version of a presentation;
- plurality of options to manage all aspects (colors, fonts, and so on);
- plurality of predefined layouts;
- possibility to manage very complex document structure;
- automatic creation of navigation elements.

3.1 The preamble

A minimal preamble for a Beamer presentation can be as follows:

```
\documentclass{beamer}
\mode<presentation>
{\usetheme{Boadilla}
\usecolortheme{albatross}}

\usefonttheme{serif}
\usepackage[italian]{babel}
% or other language
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}

\title[Short title] {Long Title}
\date{15 August 2000}
\author {Author 1 \inst{1}
\and Author 2\inst{2}}
\institute[Politecnico di Torino]
  \inst{1}
  Department of Mathematics\\
  Politecnico di Torino
  \and
  \inst{2}%
  Department of Mathematics\\
  Politecnico di Torino}

\pgfdeclareimage[height=0.5cm]{logo}
{Logo file name}
\logo{\pgfuseimage{logo}}
```

This preamble shows some characteristics of the Beamer class: "modes" and "themes".

With a unique file it is possible to obtain different outputs: the presentation to be projected, a file with the slides ready to be printed[3], an article version of the presentation, a file with talk notes.

The main file can contain instructions and text which are to be used in one or more of this "modes": in our example, instructions `\usetheme{Boadilla↩ }` and `\usecolortheme{albatross}` are contained in the command `\mode<presentation>` thus they will be used in all *modes* except article, while the instruction `\usefonttheme{serif}` will be used in all versions of the presentation.

Modes in beamer define the purpose for which the file is created:

- beamer is the default mode and it is used for files to be displayed with a projector;
- second is used when the author needs to create material to be displayed on a second screen;
- handout is used to create printed versions of the presentation (handout);
- trans is used to create transparencies;
- article transfers the control of the text to another class, the article class.

The modes all and presentation are used for content to be displayed, respectively, in all modes or in all modes except of article.

Almost all the global aspects of the presentations are defined in a *theme*.

Beamer contains global *themes*, which define every characteristic of the layout of a presentation[4]

Beamer defines also:

- *outher themes*: they control all characteristics related to the outer layout, such as header and footer, navigation bars, logos, titles;
- *inner themes*: they control all aspects of the layout of the slide content, such as frametitle, description, itemize and enumerate environments, theorem, proof, blocks environment, figures and tables and so on;
- *color themes*: they control the color palette of the presentation (`\usecolortheme{albatross}` in our example);
- *font themes*: they control the characteristics of the fonts used in a presentation (in our example, `\usefonttheme{serif}`).

Beamer, as many other packages and programs for presentations, uses as a default option sans-serif fonts.

As in every LATEX document, a title and a subtitle can be printed, as well as date and authors (with the usual instructions). Please note the instruction `\inst` which allows to declare the affiliation of an author to a University, a school or a firm.

The instruction `\logo` allows to insert the institution logo or the conference one. It is declared as a pgf[5] image.

[3] This file usually does not contain transition effects, it is less coloured and it often contains more than one slide per page.

[4] All predefined themes, both global, outer, inner, font and color, are listed in Tantau *et al.* (2015).

[5] The author of the Beamer class is also the author of the pgf package, see Tantau (2013)

3.2 The title page

A presentation usually starts with a title page, containing the title of the presentation, the authors and affiliation, the date and the conference title.

This page can be created using the \maketitle command alone, as an argument of the \frame command (or environment), or by inserting in them the \titlepage command:

```
\begin{frame}
\titlepage
\end{frame}
```

3.3 The table of contents

Beamer allows to create a table of contents by using the \tableofcontents[options] command. The table of contents is inserted in the frame in which the command appears.

The options allow to produce particular behaviours. The most useful ones are:

- currentsection makes visible only the contents of the current section and its subsections and semi-transparent the others. A similar behaviour is obtained for subsections with the option currentsubsection;
- hideallsubsections hides all subsections, while hideothersubsections hides all subsections except the current one;

All the options, along with their behaviour, can be found in Tantau *et al.* (2015) and allow a very precise control of all elements of the table of contents.

3.4 The document body

The Beamer class produces a document divided in parts, sections and subsection. This structure is used in the index of the presentation and, if used, in the navigation bar. The structure can appear in all modes or only in a particular one. For example, the code

```
\section<beamer>{This section appears ↩
    only in the beamer mode}
\section<handout>{This section exists ↩
    only in the handout mode}
```

creates sections that appear only in the mode specified in <...>[6].

Slides are created by the \frame command or using the frame environment (please note that some options are available only for the environment).

A frame always displays at least the content written in the command or environment. The other elements that can be displayed, depending on the theme, are:

- *a sidebar* for the table of contents;
- *a navigation bar* (this one is always present unless explicitly eliminated, see paragraph 3.6);
- *a bottom bar* with some information, for example author, affiliation, conference;
- *a upper bar* with the structure of the presentation, i.e. section and subsection;
- *the frame title.*

Its content can be shown all together or in different moments, maybe with effects.

For example, the following code

```
\begin{frame}
\transglitter<1-2>[direction=45]
your text here
\end{frame}
```

produces a slide with a glitter effect that sweeps in the specified direction[7].

Usually, each slide has a title and sometimes also a subtitle. There are two ways to insert a title:

- by using the command \frametitle immediately after the header of the environment:

  ```
  \begin{frame}
  \frametitle{Your title}
  your text here
  \end{frame}
  ```

- by specifying it in the frame environment:

  ```
  \begin{frame}[options]{title}{↩
      subtitle}
  your text here
  \end{frame}
  ```

The frame environment has many useful options. Among them

- the allowframebreaks option allows to split the content of a frame on more than one page (please note that this option is deprecated, as a slide should contain at most 12 lines, see section 2).
- The allowdisplaybreak option allows large formulae to split across slides;
- b, c and t manage the vertical alignment of the slide (the default is c);
- fragile allows verbatim material to be inserted in a slide;
- shrink allows to reduce the text by the specificated percentage if it is too long for a single slide (please note that this option is deprecated, as a slide should contain at most 12 lines, see section 2).

[6] The same syntax can be used in other commands to obtain the same behaviour, for example in the list commands, see paragraph 3.4.2.

[7] For other transition effects, see Tantau *et al.* (2015).

3.4.1 Overlays

The content of each frame can be displayed all together or, more often, in different steps. What we usually want is to be able to hide and display different parts of a slide. The easiest way to do this is to use *overlays* to activate and deactivate parts of a slide.

There are several ways to achieve this purpose[8]. In the commands that require the specification of the overlays, this one must be given in the optional argument `<...>`: you can specify a single overlay (`<3>`), many overlays (`<3,5,9>`), an interval (`<2-4>`), all overlays until one specified (`<-4>`) or starting from one specified (`<2->`).

The main commands for overlays are

- `\pause`: it is the simplest command. Beamer shows the content of a frame until the command and the other part of the slide are shown only when the slide is advanced by a click of the mouse or a key press;
- `\onslide<...>{text}`: it is a more powerful and flexible command, which shows `text` in the specified overlays;
- `\only<...>{text}<...>`: if one or both overlay specifications are given, `text` is inserted only on the specified slides and is thrown away in the other slides;
- `\uncover<...>{text}`: `text` is shown only on the specified slides. In the others, it is still typeset and it occupies space, but it is invisible. A similar behaviour can be achieved with the `\visible` and `\invisible` commands;
- `\alt<...>{default text}{alternative text}<...>`: if one (and only one) of the overlay specifications is given, `default text` is printed on the specified overlays and `alternate text` on the others;
- `\temporal<...>{before slide text}{default text}{after slide text}`: this command prints `before slide text` if the slides come before the one indicated in the overlay specification, `default text` on the slide indicated in the overlay specification and `after slide text` on the slides after the specified ones.

A detailed list of these commands is available in Tantau *et al.* (2015) and many examples can be found in Voss (2012).

Beamer treats the text that is not shown in an overlay in two ways: as invisible text, which occupies space and can have transparency effects, or as text to be inserted only in the specified overlays, while in the other it is somehow "thrown away". For instance, the command `\only` behaves in the second way, while the command `\onslide` usually behaves in the first way.

In Beamer many LaTeX commands and environment (such as theorem) accept as an optional command the specification of the slides in `<...>`.

In this example

```
\begin{frame}
\textbf{bold line in all overlays}
\textbf<2>{bold line only on the second ↩
    overlay}
\textbf<3>{bold line only in the third ↩
    overlay}
\end{frame}
```

the option in `<...>` specifies on which overlays the `\textbf` command should be used.

In this example

```
\begin{frame}
\begin{theorem}<1->[Lagrange]
  text of the theorem
\end{theorem}
\begin{proof}<2->
 proof of the theorem
\end{proof}
\end{frame}
```

the text of the theorem appears in all overlays, while the proof appears only from the second.

3.4.2 Lists

In Beamer you can (obviously) use the usual list environments of LaTeX. These environments have a slightly different syntax which allows to specify the behaviour of each element in a particular overlay.

It is possible to specify the overlays in which an item should be displayed with the option `<...>` in the command `\item` or specify that items are to be displayed one by one with the option `<+->` in the itemize, enumerate or description environments:

```
\begin{itemize}[<+->]
\item This item appears from the first ↩
    overlay.
\item This item appears from the second ↩
    overlay.
\item<1-> This item appears in the first↩
     overlay, as it is specified in the ↩
    option of the item itself.
\item This item appears from the third ↩
    overlay.
\end{itemize}
```

3.4.3 Highlighting of text

Beamer has different commands or environments to highlight parts of the text.

The two most useful commands are `\alert<...>{text}`, which highlights `text` by changing (usually) its color (by default in red) and `\structure<...>{text}`, which marks `text` as a part of the structure, which highlights the text in the same colour and font of other structural elements of the presentation (slide titles, navigation bar, etc.).

[8] Here we show commands to cover or uncover content in slides. Please note that the same goal can be achieved by the environment with the same or similar name (for instance, onlyenv for `\only`).

Both commands can be used in an entire environment, such as itemize or enumerate, as the following example shows (we suppose that the colour of the text is black and the color of the alerted text is red):

```
\begin{itemize}[<+-| alert@+>]
\item This item appears in the first ↩
    overlay colored in red.
\item This item appears in the second ↩
    overlay colored in red, while the ↩
    first one becomes black.
\item This item appears in the third ↩
    overlay colored in red, while the ↩
    first and second ones become black.
\end{itemize}
```

In this example, all items appear on the second overlay, but are highlighted one after the other:

```
\begin{itemize}
\item<2->\alert<2> Item 1 appears on the↩
     second overlay, in red.
\item<2->\alert<3> Item 2 appears on the↩
     second overlay, and it is red on ↩
    the third one.
\item<2->\alert<4> Item 3 appears on the↩
     second overlay, and it is red on ↩
    the fourth one.
\end{itemize}
```

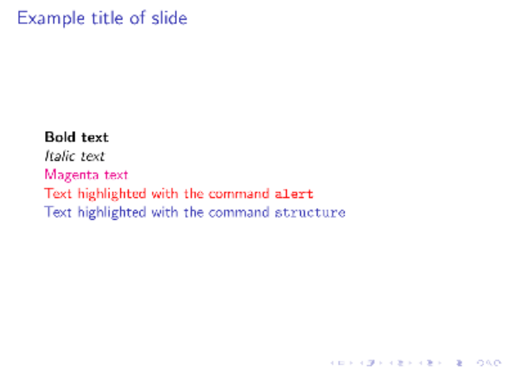
To better understand the difference between the two commands, you can see the following example. The code

```
\documentclass{beamer}

\usepackage[utf8]{inputenc}

\begin{document}

\begin{frame}

\textbf<2>{Bold text}

\textit<2>{Italic text}

\textcolor<2>{magenta}{Magenta text}

\alert<2>{Text highlighted with the ↩
    command \texttt{alert}}

\structure<2>{Text highlighted with the ↩
    command \texttt{structure}}

\end{frame}

\end{document}
```

produces a beamer presentation, the second slide of which is shown in figure 1.

You can see different types of highlighting. Please notice that the `\structure` command highlights the text in the same colour of the frame title and the navigation bar (the elements of the structure).

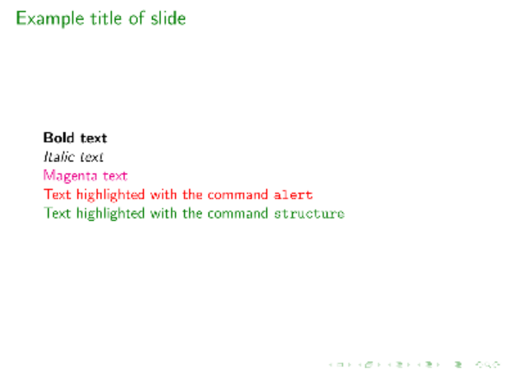
If you change the structure colour, frame title, navigation bar and argument of the `\structure` command change accordingly, as shown in figure 2.



Example title of slide

Bold text
Italic text
Magenta text
Text highlighted with the command `alert`
Text highlighted with the command `structure`

Figure 1: The second slide of the presentation with the default structure colour



Example title of slide

Bold text
Italic text
Magenta text
Text highlighted with the command `alert`
Text highlighted with the command `structure`

Figure 2: The second slide of the presentation with the changed structure colour

3.4.4 Boxes

Another way to highlight parts of the presentation is by using coloured boxes. Some environments, such as theorem, definition, example, build around the text a coloured box (by default, the first and the second use the `structure` colour, the third one uses green). It is possible to create also different boxes:

```
\begin{block}<overlays>
{header}
    text
\end{block}
\begin{alertblock}<overlays>
  {header}
 text highlighted in the alert colour
\end{alertblock}
```

It is also possible to define boxes with the desired colours or for desired purposes (for example, boxes for exercises), using the environments beamercolorbox and beamerboxesrounded:

```
\setbeamercolor{postit}{fg=black,bg=↩
    yellow}
\begin{beamercolorbox}[sep=1em,wd=5cm]{↩
    postit}
text
\end{beamercolorbox}

\setbeamercolor{uppercol}{fg=white,bg=↩
    green}
\setbeamercolor{lowercol}{fg=black,bg=↩
    green}
```

```
\begin{beamerboxesrounded}[upper=↩
    uppercol,lower=lowercol,shadow=true↩
    ]{Theorem}
$A = B$.
\end{beamerboxesrounded}
```

3.4.5 Verbatim mode

Verbatim material in beamer needs a special treatment. A slide which contains verbatim must be declared with the option fragile. This option tells that the verbatim code must be written on an external file which will be read back in order to treat the material correctly:

```
\begin{frame}[fragile]
verbatim text
\end{frame}
```

Another way to insert verbatim text in a frame is to use the semiverbatim environment:

```
\begin{frame}
\begin{semiverbatim}
verbatim text
\end{semiverbatim}
\end{frame}
```

3.4.6 Figures

Figures can be included in a beamer presentation in two ways:

- with the usual command \includegraphics, which has (as almost all commands in beamer) an extended syntax:

  ```
  \includegraphics<overlays>[↩
      settings]{file name}
  ```

  which allows to specify also the overlays in which the image must appear;

- with the couple of commands (from the pgf package) \pgfdeclareimage and \pgfuseimage:

  ```
  \pgfdeclareimage[settings]{beamer ↩
      name}{file name}
  \pgfuseimage[settings]{beamer name↩
      }
  ```

  In this case, the overlays in which the image must appear are set with one of the commands seen in paragraph 3.4.1.

This is an example of the two commands in a slide:

```
\begin{frame}
\includegraphics<1->{image1}%%% this ↩
    image appears on all overlays
\includegraphics<2->{image2}%%% this ↩
    image appears from the second ↩
    overlay
\end{frame}
\pgfdeclareimage[scale=0.5]{image3}{↩
    images/image3}
\pgfdeclareimage[scale=0.7]{image4}{↩
    images/image4}
\uncover<3->{\pgfuseimage{image3}}%%% ↩
    this image appears from the third ↩
    overlay
\only<4>{\pgfuseimage{image4}}%%% this↩
     image appears only on the fourth ↩
    overlay
```

3.4.7 Time settings

Beamer offers the possibility to set the duration of each overlay. This purpose can be achieved with the command \transduration:

```
\transduration<overlay specification>{↩
    number of seconds}
```

For instance, the code

```
\transduration<1>{10}
```

sets the duration of the first overlay to 10 seconds, then the overlay is changed to the second.

3.4.8 Sounds and animations

A very interesting possibility offered by beamer is to magnify parts of a very complicated slide or very big figure (zoom effect) with the command \framezoom:

```
\framezoom<button overlay specification↩
    ><zoomed overlay specification>[↩
    options]
(upper left x,upper left y)(zoom area ↩
    width,zoom area depth)
```

For example, the code, taken from Tantau *et al.* (2015)

```
\begin{frame}
\frametitle{A Complicated Picture}
\framezoom<1><2>(0cm,0cm)(2cm,1.5cm)
\framezoom<1><3>(1cm,3cm)(2cm,1.5cm)
\framezoom<1><4>(3cm,2cm)(3cm,2cm)
\pgfimage[height=8cm]{↩
    complicatedimagefilename}
\end{frame}
```

produces three zooming areas for the big picture, each one in a different position and to be displayed on different overlays, starting from the second.

It is also possible to insert into the presentation multimedial files, both audio and video. Please note that the files will not be inserted in the pdf file, which contains only a link to the audio or video file. Thus it is necessary to have multimedial files with the presentation in order to display them. Depending on the PC configuration, the file will be displayed into the presentation or in an external viewer. Audio files can be loaded into the presentation with the command

```
\sound[options]{sound poster text}{sound↩
     filename}
```

and videos can be loaded with the command

```
\movie[options]{poster text}{movie ↩
    filename}
```

from the multimedia package.

The command

```
\animate<overlay specification>
```

allows to create animations by showing overlays in rapid succession.

Settings of the animation can be modified with the command

```
\animatevalue<start slide-end slide> {↩
    name}{start value}{end value}
```

For example, if you create a sequence of pictures with name animate1, animate2, ..., animate10, the following code, taken from Tantau *et al.* (2015)

```
\begin{frame}
\animate<2-9>
\multiinclude[start=1]{animate}
\end{frame}
```

allows the creation of a simple animation by displaying the images in rapid sequence. The command \multiinclude allows to load all images with name animate followed by a number.

For a more detailed description of the use of animations in beamer, see Pignalberi (2010).

3.5 Fonts

In beamer, fonts are managed through *themes*, as described in paragraph 3.1.

The command \usefonttheme{default} loads a sans serif font for all the presentation. Some character glyphs in mathematical text are replaced by more appropriate versions automatically. This produces a text with glyphs from two different collections, which gives sometimes strange results.

Claudio Beccari has created the lxfonts package to overcome the problems of the standard font used for slides. The package and its usage are fully described in Beccari (2007) and in Beccari (2013).

To use this fonts in beamer it is necessary to change the font theme from default to professional:

```
\usefonttheme{professionalfonts}
```

This command tells beamer not to make any substitution, as it is managed by the font package.

The package is then loaded by the usual command

```
\usepackage{lxfonts}
```

Please note that the package should be loaded after having loaded all the other fonts packages.

A demo of the results of the package is contained in the documentation, see Beccari (2013).

3.6 Navigation bar

The majority of beamer templates has a *navigation bar* in the bottom right corner of each slide, as shown in figure 3.

The symbols, from left to right, are:

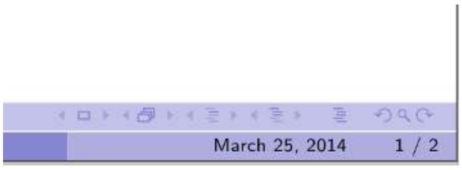


Figure 3: Beamer navigation bar

- the *overlay* icon, a single rectangle with forward and backward arrows to navigate from an overlay to the others of each slide;
- the *slide* icon (or *frame* icon), a set of rectangles with forward and backward arrows;
- a *subsection* icon, a highlighted line in a symbolic table of contents, with forward and backward arrows;
- the *section* icon, a highlighted line with smaller lines below for subsections, with forward and backward arrows;
- the *presentation* icon, a highlighted symbolic table of contents;
- the *search* icon, a magnifying glass with forward and backward arrows.

If these symbols are not needed, they can be disabled with the code

```
\setbeamertemplate{navigation symbols}{}
```

If other symbols are needed, they can be added with the command \insert with the name of the symbol attached. For example

```
\insertframenavigationsymbol
```

allows to add the icon to navigate slides backward and forward.

3.7 Multi-column layout

When building a presentation, it may be useful to put the content of a slide on more than one column. For instance, this layout can be useful to insert into the slide a figure and its explanation on the right side.

The code for building a multi-column layout is the following

```
\begin{columns}[settings]
\begin{column}[position]{width}
  content of first column
\end{column}
\begin{column}[position]{width}
  content of second column
\end{column}
...
\end{columns}
```

This syntax makes all columns appear on the first overlay of the slide. If you need to make them appear one by one, you can use the commands described in paragraph 3.4.1, for instance \only.

3.8 Advanced personalization: new commands and environments

Beamer offers a wide number of ways to personalize a presentation: it has commands for modifying the layout, the theme, and so on.

Over the internet, you can find a lot of personalized templates, you should only search for “beamer templates” on every search engine.

If you want to create your personalized theme, you can find a good introduction to personalization in Fiandrino (2014).

Here we want to show how to create (or redefine) commands or environment in beamer in order to make them aware of overlays. We have seen all along the paper that a lot of commands are redefined to be visible in some overlays and invisible in other (see paragraph 3.4.1 for some examples).

When defining or redefining a command or environment, you can specify in the definition that it can have a different behaviour on different overlays:

```
\newcommand<>{command name}[argument ↩
    number][default value]{text}
\renewcommand<>{existing command name↩
    }[argument number][default value]{↩
    text}
\newenvironment<>{environment name}[↩
    argument number][default value]{↩
    begin text}{end text}
\renewenvironment<>{existing ↩
    environment name}[argument number↩
    ][default value]{begin text}{end ↩
    text}
```

The difference with the standard similar commands of LaTeX is that here the number of parameters accepted is equal to argument number plus one, being the latter the overlay specification for the command or the environment to operate.

For instance, the code

```
\newcommand<>{\makemegreen}[1]{{\color↩
    #2{green}#1}}
```

produces a text which is coloured in green on the specified overlays and in the normal colour on the others.

Other examples can be found in Tantau *et al.* (2015).

4 Other classes to write presentations

We have shown the main features of the beamer class to build presentations, as it is a widely used tool for this purpose.

There are nevertheless many other classes that can be used. Among the others, we note

- overlays: it allows to write presentations with incremental slides;
- gridslides: it allows to create free form slides with blocks placed on a grid. The blocks can be filled with text, equations, figures and so on;
- texpower: it is a bundle of packages that provide an environment for creating pdf screen presentations;
- ffslides: it is a small set of macros added to the article class, with the aim to easily design documents such as presentations, posters, research or lecture notes, and so on;
- ifmslide: it is used to produce printed slides with LaTeX and online presentations with pdfLaTeX;
- powerdot: it is a presentation class for LaTeX that allows for the quick and easy development of professional presentations;
- other classes or packages: lecturer, pdfslide, talk, prosper, fancyslides, elpres, ppower4.

Powerdot is well described in Voss (2012); a wide range of examples are available in `https://www.ctan.org/pkg/presentations-en`.

References

Beccari, Claudio (2007). «I font per le slide LaTeX resuscitati». *ArsTeXnica*, (4), pp. 82–87. `http://www.guitex.org/home/numero-4`.

— (2013). «Lxfonts – set of slide fonts based on cm». `CTAN:fonts/lxfonts`.

Fiandrino, Claudio (2014). «Introduzione alla personalizzazione di beamer». `http://www.guitex.org/home/images/doc/GuideGuIT/intropersbeamer.pdf`.

Pignalberi, Gianluca (2010). «Presentazioni animate in LaTeX». *ArsTeXnica*, (10), pp. 33–40. `http://www.guitex.org/home/numero-10`.

Tantau, Till (2013). «The TikZ and PGF Packages». `http://sourceforge.net/projects/pgf`.

Tantau, Till, Joseph Wright and Vedran Miletić (2015). «The beamer class». `CTAN:macros/latex/contrib/beamer/doc/beameruserguide.pdf`.

Voss, Herbert (2012). *Presentations with LaTeX*. Lehmanns media, Berlin, 1st edition.

▷ Grazia Messineo
Università Cattolica Milano – IIS “Falcone-Righi” Corsico
`grazia dot messineo at unicatt dot it`

▷ Salvatore Vassallo
Università Cattolica Milano
`salvatore dot vassallo at unicatt dot it`

The **Toptesi** package
Typesetting a PhD thesis with LaTeX

Claudio Beccari

Abstract

This tutorial uses the information given in the previous five ones in order to describe how to use the TOPtesi LaTeX package to typeset a PhD thesis. This package has a specific option to configure the typesetting of such a kind of thesis in the format agreed upon by ScuDo, the doctoral School of Politecnico di Torino.

Sommario

Questa lezione raccoglie l'informazione fornita dalle cinque precedenti al fine di descrivere l'uso del pacchetto TOPtesi per produrre con LaTeX una tesi dottorale. Questo pacchetto dispone di una specifica opzione per configurare la tipocomposizione della tesi nel formato concordato con la ScuDo, la Scuola di Dottorato del Politecnico di Torino.

1 Introduction

The TOPtesi LaTeX package (Beccari, 2019c,a)[1] to produce theses of different levels has been around for many years; the successive previous versions started to become too complicated and cumbersome. Now it has a modular structure and a suitable module is selected by expressing options in the form of *key* = *value*; in particular the type of doctoral thesis to be typeset at the Doctoral School of Politecnico di Torino is selected with the *key* = *value* pair set to tipotesi=scudo. The doctoral thesis style and structure are specific for ScuDo; several other options select other styles for other thesis types with other structures.

2 The class structure

Figure 1 shows the various modules that form the TOPtesi bundle. Each module is a file by itself and options are specified to select which module to use in order to typeset the desired title page and to configure the preamble of the document so that the thesis fulfils its requirements.

The user can specify his/her preferred packages; not any package, because it is important to avoid conflicts with the bundle modules and settings; but there is an ample choice.

Notice that the input source file `.tex` is fed to a toptesi.cls class, that receives the options and passes them to the selected modules. The first one is toptesi.sty; why another package when the class could do everything is desired? Simply because any user can choose to employ the `.sty` package with a different class, for example book.cls, or report.cls, or any other compatible class.

Myself, as the author of TOPtesi, I think it is not worthwhile to do such a mixture, but my opinion is evidently biased. I just want to emphasise that the possibility does exist, but unusual classes might form an unusual couple with toptesi.sty, and might perform in a bad way. I cannot tell which classes are compatible with toptesi.sty; I just can say that report.cls is the base class on which toptesi.sty works by default. Compatibility exists with book.cls, but I did not made any tests with other non standard classes.

Evidently the main class and the main package, with the possible support of the user loaded packages, and possibly with the further bundle topcoman.sty module, cooperate to the form of the output `.pdf` file typeset contents.

The various *values* assigned to the tipotesi key, select eight different modules to typeset eight different thesis types with different title page arrangements and internal structures.

The topfront.sty module is maintained for backwards compatibility; but this module is selected if no other module has been chosen (no option or no value specified) or when a wrong or misspelt *value* has been specified. Chances are that the compilation ends correctly, but if the user gives an attentive look to the result s/he might notice, for example, wrong hyphenation or different labels on the pieces of information that have been input, for example a high school name in place of a university name. But these occurrences should ring a bell to the user in order to inform him/her that some option value is wrong.

3 The option values

Let us describe the eight options

tipotesi=triennale Prepares a title page suited for a bachelor degree final work and a suitable structure; no extra packages are loaded besides the few ones that shall be described in

[1] The bibliography at the end of this paper contains many references to bundles and packages shipped with any TeX system complete installation, be it TeX Live or MiKTeX. The references that carry the notice "Readable with..." are all already in the users' computer where the TeX system is installed so no Web search is necessary.

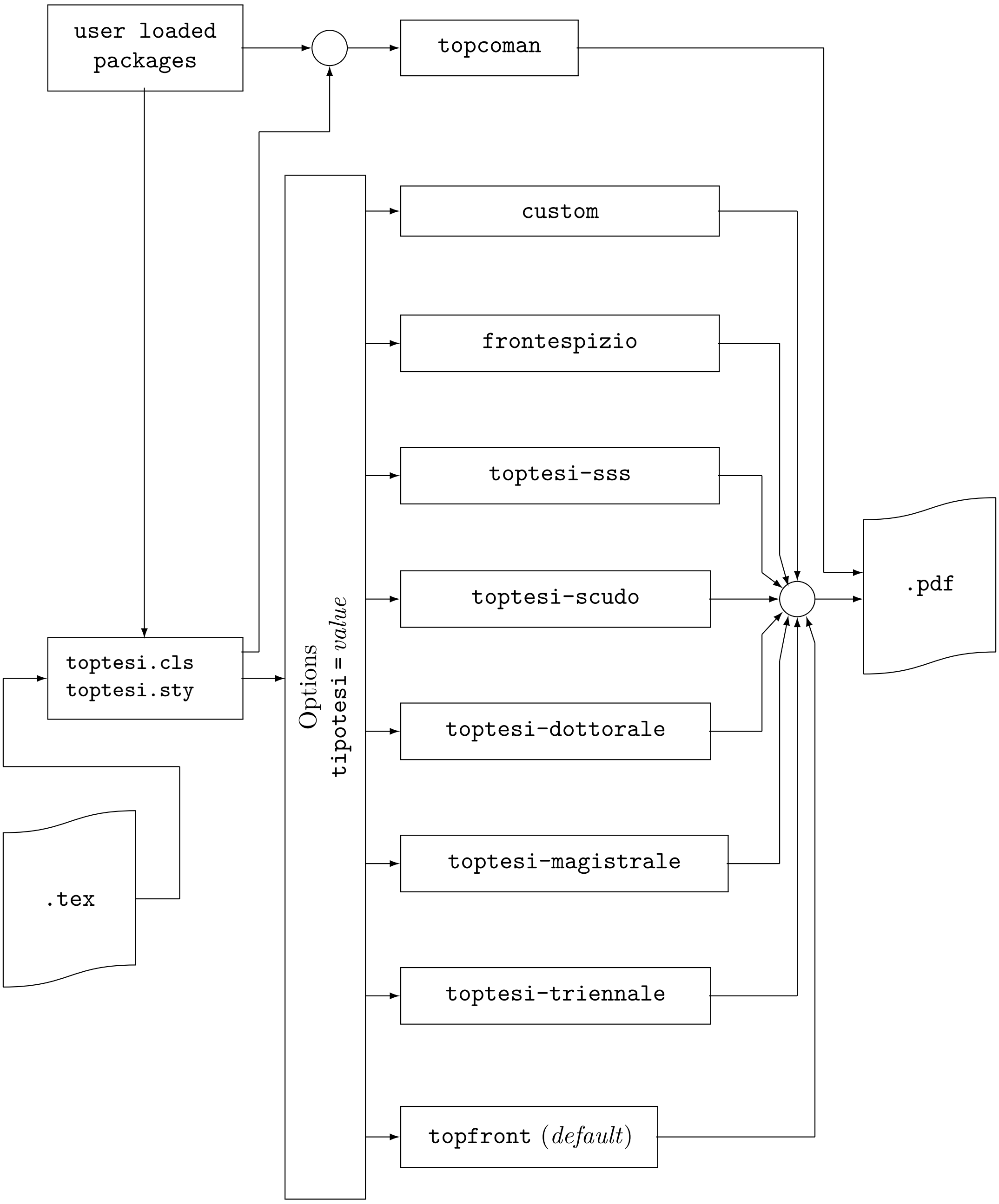


Figure 1: Version 6.x TOPtesi bundle flow diagram

the following sections. The default language is Italian, but with the use of the declarations `\english` or `\italiano` (or with suitable groups or environments) it is possible to switch back and forth between these two languages.

tipotesi=magistrale Prepares a title page suited for a master thesis and a suitable structure; no extra packages are loaded, as with the bachelor degree option and, similarly, it is possible to switch back and forth between Italian and English.

tipotesi=dottorale Prepares a title page and a suitable structure valid for doctoral theses in general. This option should be suitable for PhD students in other doctoral Schools. Language switching is similar to the previous thesis types. No extra packages are loaded.

tipotesi=scudo Prepares a title page and an internal structure suitable with dissertations to be defended after frequenting the Doctoral School of Politecnico di Torino. Some other packages are preloaded, and the *default language is English.*

tipotesi=sss This unusual and unique option is used to typeset a high school final work. This module was asked for by high school students who were already familiar with LaTeX. Unfortunately the Italian Ministry of Education and the Parliament decided to abolish the final work report and this year 2019 is the first year where the final high school state exam does not require the preparation of a such report. The module is conserved for backwards compatibility.

tipotesi=frontepsizio This option excludes the internal module topfront to be used to typeset the title page, but it loads the frontespizio package that is alternative to this bundle module since they are mutually incompatible. This option does not forbid the user to employ the alternative package, but the option must be specified in order to avoid loading what might be in conflict with the chosen package.

tipotesi=custom This option leaves the user completely free to compose the title page "by hand", that is the user can use a titlepage environment, where s/he can put whatever piece of information, typesetting it with any font of any size, any family, any series and any shape available for that font. The structure of the document remains the default one and almost any extension package chosen by the user can be loaded.

tipotesi= No option or no value or a wrong value is specified; this is maintained for backwards compatibility and as a fall back style in case of errors.

This variety of choices requires a thick documentation; the file `toptesi.pdf` and `toptesi-it.pdf` are part of the bundle; they are directly accessible through the `texdoc` terminal command with the following syntax[2]:

```
texdoc toptesi.pdf
texdoc toptesi-it
```

The first text is completely in English, but most of it is the documentation of the code; in spite of this, when the codes of the various modules are commented, there are many explanations, suggestions and examples of how certain commands may be used, or certain solutions may be tweaked, or certain strings may be adjusted to the users' needs.

The second text is mostly in Italian, but the parts that are connected to the tipotesi=scudo option are in English; it deals mostly on how to compose the various thesis types and it explains why there are so many differences in the title pages.

4 General information for various thesis types

The structure of a thesis may vary according to its level and to the scientific domain it is about. Let us deal only with the three university levels: bachelor, master, and doctoral thesis types.

4.1 Bachelor final work or bachelor thesis

In some countries and in some universities the bachelor university courses are completed with a final work that generally deals with what has been learned during the degree course: it is "just" an application of what the candidate did actually learn during his/her studies. No one expects that the final work contains new theoretical aspects. The value of the final work is measured through the quality of the application. Sometimes this final work is labelled as bachelor thesis; actually it is not excluded that something new appears in such a thesis, but in general it is a smart application of known practices.

This has an influence on the structure of the thesis; in the sense that initial chapter(s) on the state of the art are usually missing, and the same is valid for the final chapters: conclusions, further investigation, possible developments, and the like.

Therefore this type of document contains the title page, seldom it contains a legal page, certainly it contains the table of contents, and possibly the list of figures and that of tables. There will be an introductory chapter that describes what the whole thesis is about and what was its purpose; one or more chapters describing what has been done; such chapters, depending on the scientific domain, might be rich of figures, drawings, photographs, and the like; they will contain tables of measures, or material specifications, or tables of temporary

[2] In the first case the extension is compulsory, because without it the second file is opened.

or final results, or similar tables; if the thesis deals with the design of an equipment, of a machinery, of a plant, or a building, the executive drawings may require large sheets of paper, folded in a particular standard way, or outside the thesis collected in cardboard tubes.

Some of this material may be created with LATEX, but some, especially large drawings, require special drawing equipment and/or special software and special printers. Here I will not deal with the special resources external to the typesetting procedures.

Text and tables are done in LATEX; the latter are more difficult to typeset, because it is difficult to preview what is actually desired to obtain.

4.2 Master thesis

Master theses are real theses. They are supposed to be defended in a formal examination session where the candidate should describe his/her work and defend it against the objections of the examining committee. This type of examination is the traditional one that ends with a laurel wreath (*corona laurea* in Latin) to be worn by the new Master. Several mottos contain the word, in particular that of the late Institution of Electrical Engineers (IEE, now merged with the IET): «Lauream ferat qui meruit». The academic title in Latin is “Magister”, the word where the name “master” comes from.

A master thesis should contain something really new, some new theory, or some demonstration that some conjecture is true. Sometimes a very good master thesis may receive the declaration of being worth of publishing.

Evidently the introductory part and the concluding part of a master thesis are wider and more detailed than the corresponding parts of a bachelor thesis; nevertheless it is not the number of pages the element that distinguishes a really good thesis; among the glories of Politecnico di Torino there is an engineer who stated and proved the theorem of virtual works, now known as the Theorem of Castigliano; Castigliano’s thesis did not go over thirty pages.

The typographical elements of a master thesis are more or less the same as those of a bachelor thesis. The title page is different and the legal page is often required.

4.3 The doctoral dissertation or PhD thesis

Similarly, the doctoral dissertation is an even more complete work of research than the master thesis; very often it reports on a research that the candidate developed during his/her study period on a PhD degree course, and partial results have already been published on scientific journals.

The Latin academic title is “Doctor”, where the modern title comes from, including the adjective “doctoral” that qualifies the thesis, often called dissertation.

The typographical elements of a doctoral dissertation are not very different from those already described for the other theses. The title page might contain a lot of information concerning the formal examining commission.

The disciplines dealt by a doctoral dissertation are the most varied and go deep into the details. Every scientific discipline has its jargon and its habits. It is possible to distinguish a thesis in hard disciplines from those in soft ones by looking at the footnotes and bibliographies; the body of the text may contain a lot of mathematics, or a lot of verbatim citations of other authors. Here comes the utility of a specific LATEX class in order to properly typeset these elements.

4.4 The thesis bibliography

All thesis types contain a bibliography; single references in this bibliography are cited again and again in the text body.

There are many ways to typeset bibliographies and to cite their references. LATEX offers several packages and programs to handle these typographical units.

The best way to handle a bibliography is to write a bibliographic database where each record describes a reference in a way suited to its nature: a book, a report, a chapter, a contribution to a collective book, and article in the proceedings of a conference, an article in a journal, another thesis, and so on; each type of reference may have different elements: title, author(s), publishing house with its location, volumes of a collection of proceedings, dates, conference name and location, and so on. Such a database is precious; but is not in a form that LATEX can directly handle in order to produce the references in a recognised style and the citations in the form needed by a specific discipline. LATEX provides several bibliographic style files and the TEX system installation has at least two main programs that extract the necessary information from the bibliographic database and feed such information to LATEX in a form that allows it to typeset the bibliography with the proper style.

5 Lists and tables

Lists are of three main groups: itemising, enumerating, descriptive lists. The items listed into an itemising list are marked with a not alphabetic symbol that is equal for all items; more often than not this symbol is a bullet.

Enumerations number their items with ordered symbols, typically arabic numbers, upper- or lowercase roman numbers, lower- or uppercase letters of some alphabet. It is generally possible to label one or more items of an enumeration so as to recall them for reference purposes.

Descriptions are similar to itemisations, where each item is introduced by a word or a short phrase that is being described in the item body; but, differently from itemisations, each item may consist of one or more full paragraphs.

In technical writings itemisations are part of the paragraph that introduces them; therefore items start with a lower case initial first word and never end with a full stop, except possibly the last item in the list. This implies that the list items cannot contain more than a fraction of a paragraph; rather it contains simple sentences, or simple phrases, or single words; punctuation at the end of each item may be made with commas and semicolons, never periods or other terminal punctuation signs; the item final punctuation may be omitted in a displayed itemisation.

On the opposite, enumeration and description items form paragraphs and start with an uppercase letter and end with a full stop; such items may contain more than a single paragraph.

These details must be taken care of by the user, they cannot be handled in an automatic way by LaTeX.

Tables may be of different types; they may be small or large, remain within the margins of the printed block, or be wider; they may be in a normal position or rotated 90° counterclockwise; they may occupy one page or may be several pages long.

For each of these tables LaTeX has various solutions and it should be the user responsibility to know what to do to handle a specific table and/or which package to load that has a ready to use solution.

In all cases the user should plan a table very carefully; yes *plan*, because a table is not simple text, where LaTeX can do a beautiful job for splitting a sequence of words or similar entities into lines of equal measure while minimising the inter word space. Tables are bidimensional objects and must be handled in a different way; LaTeX can do many things except typesetting in a beautiful way a badly planned table.

6 Images of any kind

LaTeX, as a mark up language, has all the necessary commands for both drawing certain images and importing certain types of already made images. In any case it is impossible to produce any image in a LaTeX document simply by dragging a graphic file from a folder to the editing window, as it is possible to do, in certain cases, with some word processors. Let us separate these two operations.

Importing images This is relatively simple because the fundamental command `\includegraphics` is self explanatory. The point is how to use correctly the various options accepted by this command, and which kind of images the command can include; for what concerns the options, they will be discussed further on; for what concerns the images, all typesetting engines, except the simple `latex`, can import images in the formats (*a*) EPS, PDF, and MPS, that are or may be of vectorial nature; and (*b*) JPG and PNG, that refer to bitmapped formats; such image files may be imported if and only if they are well formed and contain the metadata that specify their characteristics, in particular their natural dimensions; otherwise it is necessary to include such dimensions within the options to the `\includegraphics` command. This method can be used also for other bitmapped image formats, but it is often difficult to discover the correct image natural dimensions. In any case such bitmapped images cannot be scaled at will: if they get downscaled they might loose definition; while if they are upscaled, their granularity becomes very evident: see figure 2 to notice the difference with a bitmapped character and a vectorial counterpart, both enlarged by a very large scaling factor.

Drawing images Any TeX system installation contains several packages to make drawings; the native LaTeX environment picture, that can be extended by loading package pict2e ((Gässlein *et al.*, 2016); this extension was already documented by Leslie Lamport in the second edition of his LaTeX manual, (Lamport, 1994)); a larger extension is obtained by loading package curve2e, (Beccari, 2019b), that extends pict2e — figure 1 was drawn within a picture environment extended with package curve2e. Beautiful drawings may be created with the powerful packages TikZ (Tantau, 2019) and pgfplots (Feuersänger, 2018); or with the even more powerful package PSTricks; even with METAPOST, (Hobby, 2018); with the interactive interface `asymptote` (Hammmerlindl *et al.*, 2018); with the external program `gnuplot` (Miklavec, 2013).

Furthermore it is possible to download from the Internet other programs that are LaTeX aware. The cited *packages* allow to execute "programmed drawings" while the cited *programs* generally work in an interactive mode so that the user works with his/her mouse instead of writing code. The subject is too large to be described here even if some information has been given in the previous tutorials. For what concerns the packages and the programs included in any complete TeX system installation, the documentation may be read by simply typing in a command window `texdoc` followed by the name of the

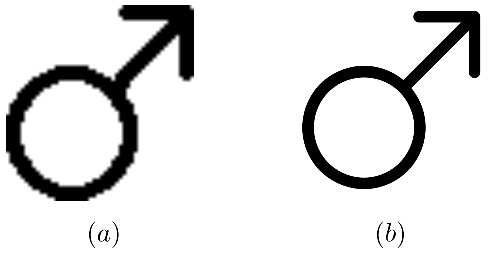


Figure 2: Comparison between an enlarged bitmapped charcter and its vectorial equivalent

> package or of the program; this is what has been recalled in the bibliography of this paper.

It is evident that if vectorial drawings are available their rendering is much better than with the bitmapped ones. For photos the lossy compressed bitmapped JPG format is acceptable, provided that the pixel density (pixels per inch) is not too small: a density of 150 pixels per inch is generally good, but remember that the graphic file quadruples its size if the pixel density is doubled; the user must seek a compromise between pixel density and quality of the rendering. The lossless compressed bitmapped PNG format is better for line drawings compared to the JPG one, but the compression is less effective than with JPG drawings; at the same time the lossy nature of the JPG format may degrade line art by superimposing displaced phantom replicas of the main drawing.

Vectorial formats are preferable, but beware: a vectorial file, such as EPS or PDF, may contain bitmapped images; therefore control very attentively before using such a "fake" vectorial image; such control is very simple: open the vectorial image to be tested and enlarge it at least by a factor of 10: if the quality of the image does not change it is truly vectorial, otherwise it is a vectorial container of a bitmapped image.

7 Mathematics

The mathematics of hard sciences (experimental sciences) deal with *quantities*, i.e. symbols that represent the pair *measure plus unit of measure*, rather than mathematical variables; it is fundamental what has been described in a previous tutorial on typesetting mathematics and on using package siunitx (Smith, 2018).

Nevertheless there are some international regulations, (ISO-31/XI, 1978), that require a specific usage of math fonts for physics and technology; in general mathematics are typeset with special characters that are classified with the name of *math groups*; these include the operators, letters, symbols, and extensible delimiters groups; special documents may require other groups. The latter two groups are self explanatory; the former ones may be confused with the roman and the italic fonts; they are not to be confused, because even if some of the glyphs appear identical, their properties are different. With the letters group the difference is very noticeable: the phrase *different affinity* (written with a text italic font) becomes $differentaffinity$ with the letters group, and it becomes *differentaffinity* with italic font used by the `\mathit` command. The lack of ligatures and inter-word spaces is evident with the letters group; with `\mathit` the ligatures are still there, but the inter-word spaces are still missing.

The ISO regulations (ISO-31/XI, 1978) in general require specific usage of bold face series; roman, italic, sans serif upright, and sans serif oblique have special meanings, the details of which may be found in the mentioned regulations and in other texts; I would suggest the freely downloadable manual (Thompson and Taylor, 2008), that explains how to use units of measure and how to write mathematics; it also adds several pieces of information that are very difficult to find elsewhere.

8 Nomenclature and glossaries

Strictly speaking a nomenclature list or a glossary are not needed in a thesis of any type. But in doctoral theses, that deal with research, a nomenclature, or glossary, or acronyms list may be useful.

LaTEX offers several packages to typeset such lists; the most common packages are the nomencl (Veytsman *et al.*, 2019), glossaries (Talbot, 2019), and acronym ones (Oetliker, 2015). It should be stressed that such packages may compose several such lists, but each one is specialised in one of them; they require the use of an external program to sort and format each entry; more is said below in connection with the ScuDo doctoral theses.

9 Indices

Again indices are not necessary in any thesis type. Nevertheless the doctoral theses, due to their advanced contents, may benefit from the presence of one or more indices.

Here we suggest the imakeidx package by Gregorio (2016) because it can solve all the problems of index production; it is highly configurable so that it is possible to produce indices with a specific name different from the default one and from the names of other indices in the same document; it is possible to decide to compose in one, two, or three columns, and a specific index style for the entries, plus a nice series of other customisations. The tipotesi=scudo option preloads this package.

The documentation of imakeidx (Gregorio, 2016) explains not only the usage, but offers a variety of examples and various tricks to tweak the entry style.

10 Archivable format

Most universities require every student to submit a file containing his/her thesis to be archived for legal documentation and possible information for other people; Politecnico di Torino is one of them.

Any archivable document must fulfil the requirements stated by specific ISO regulations. Such regulations have been published and updated several times, (ISO 19005-1, 2005; ISO 19005-2, 2011; ISO 19005-3, 2012); they are labeled by acronyms such as PDF/A-1a, PDF/A-1b, and so on.

All regulations require the PDF format; they require also certain requirements on imported pictures and their color profiles; they require that the used fonts are embedded in the file, at least the subsets of the used characters, and that the fonts be encoded according to the UNICODE standard; no character in any font should have a vanishing width. Moreover some of these regulations require that the contents of the file be of the kind known as *Tagged PDF*.

As of today, the TeX system typesetting programs based on the LaTeX mark up cannot produce tagged PDF files; in this very GuIt Meeting another speaker describes the work that is being done in order to introduce this feature into the output PDF files. Possibly in the near future all the programs that use the LaTeX mark up, at least LuaLaTeX, will be capable of producing tagged PDF files; LuaLaTeX is also convenient in order to handle UNICODE encoded fonts.

pdfLaTeX cannot be used because it can handle only one-byte encoded fonts; the first "page" of the UNICODE standard contains only the ASCII one-byte encoded subset, but this subset does not contain either accented characters, or accent glyphs to superimpose to the unaccented ones; therefore pdfLaTeX is out of the game.

XeLaTeX, also, is out of the game; it is not impossible to use it, but in order to end with an archivable format it is necessary to do some post-processing and to accept some compromises.

Therefore only LuaLaTeX is effectively suitable for producing PDF/A files, at the moment just non tagged PDF ones. Therefore the ISO standard obtainable is the one labeled PDF/A-1b.

Some universities would like to have archivable documents accessible to impaired readers; the official ISO regulations for this task are still under discussion, but work is being done also in the TeX world, in particular by the Team on Accessibility at the University of Turin. This Team has already produced a package suitable for use by blind or impaired vision readers (Ahmetovic *et al.*, 2018); work is still in progress.

11 Comments

The preceding sections mostly describe the typesetting problems that are common to any thesis type. Now it is time to concentrate on PhD theses, in particular those that are conformant with the requirements of the ScuDo Doctoral School of Politecnico di Torino.

12 TOPtesi and its tipotesi=scudo option

What has been described in section 4 about thesis types in general applies also to those to be developed at the ScuDo Doctoral School unless something special is unique for the latter ones.

The specific option tipotesi=scudo, to be specified to the toptesi class, configures the title page, the copyright or legal page in the proper way, and preloads a number of packages that are considered essential for doctoral theses in the scientific domains that form the various PhD degree courses in the ScuDo Doctoral School.

All or most of these degree courses find their counterparts in the international Institutions that deal with engineering professions, including some professions dealing with architecture. I mention only the Institution of Electrical and Electronics Engineers (IEEE) simply because it is closer to my scientific interests, but this is not a selective choice.

12.1 Facility for using either pdfLaTeX or LuaLaTeX

With the tipotesi=scudo option the preamble of the main source file is suitable to detect which typesetting program is being used: pdfLaTeX or LuaLaTeX. The former is perhaps the most suitable to use when working on the thesis drafts before the final version; the latter is suitable for producing the final PDF/A compliant PDF file.

Actually it is up to the user to configure the preamble for this task: the TOPtesi bundle contains a `toptesi-scudo-example.zip` file that has a complete commented example that is worth studying in detail. Neglecting the comment lines, this example file preamble contains the following lines:

```
\documentclass[%
  corpo=12pt, % optional;
              % default font size:= 10pt
  twoside, % recommended
  tipotesi=scudo,
  mybibliostyle, % necessary only if
                 % bibliography is typeset
                 % with different style
  numerazioneromana,% roman page numbering
                 % just for testing
                 % don't use in your thesis
]{toptesi}

```

```
14 \ifPDFTeX
15   \usepackage[utf8]{inputenc}%
16   \usepackage[T1]{fontenc}
17 \fi
18 ...
19 \ifPDFTeX % using pdflatex
20   \usepackage{lmodern} % Default
21   %\usepackage{newtxtext,newtxmath}%
22       % Times eXtended for text and math
23   %\usepackage{fourier}% Utopia,
24   %    Helvetica and "monospace = ?"
25 \else % using lualatex (or xelatex)
26   \usepackage{fontspec}
27   \defaultfontfeatures{Ligatures=TeX}
28   \setmainfont{Libertinus Serif}
29   \setsansfont{Libertinus Sans}
30   \setmonofont{Libertinus Mono}%
31               [Scale=MatchLowercase]
32   \usepackage{unicode-math}% add special
33            % math style option here
34            % for example [math-style=ISO]
35 % define one math font
36     \setmathfont{Libertinus Math}%
37 \fi
38 ...
39 \makeindex[intoc]% collect material
40                  % to one index and list
41                  % the index in the
42                  % table of contents
43 ...
44 \ifmybibstyle % customise bibliography
45   \usepackage[autostyle]{csquotes}%
46                % necessary for biblatex
47   \usepackage[backend=biber,
48               style=philosophy-classic,
49               scauthors=all,
50               sorting=nyt,
51               natbib]{biblatex} % LaTeX
52               % specific bibliography
53               % handler
54   \addbibresource{\jobname.bib}%
55            % bibliographic data base(s)
56 \fi
57 ...
58
59 \ifPDFTeX \usepackage{indentfirst}\fi
60 \raggedbottom
61
62 \begin{document}
```

This (partial) preamble needs some comments.

1. The lines that contain only three dots, denote skipped material that is not worth commenting here; they mostly deal with variants required when the PDF/A version of the thesis is required.
2. Lines from 1 to 11 show the document class statement with the options; please notice that the last two options are discouraged; option numerazioneromana is for using roman numerals in the front matter; actually this style of numbering is not necessary in modern documents; it was necessary when typographies worked with metal type. In some disciplines this traditional numbering is still appreciated, but there is absolutely no need to use it simply because it is available; use it to test the result, but avoid using it in your real thesis.
3. The mybibliography option is discouraged because in scientific publications dealing with the disciplines connected with engineering, the concise numerical style is highly preferred. Nevertheless some disciplines (or some PhD students) prefer the author-year style or some other style, so they can implement their choice, but they have to explicitly use a code similar to the one shown in lines from 43 to 55.
4. The various tests produced by the `\ifPDFTeX` conditional command detect if the thesis is being typeset by using pdfLATEX; else LuaTEX or XƎLATEX is assumed, even if XƎLATEX is discouraged as explained before.
5. In lines from 13 to 16 the necessary encodings are specified if pdfLATEX is being used; it is not necessary to specify such encodings with LuaLATEX because UNICODE (or UTF-8, UNICODE Transformation Format) is assumed for both input and output.
6. Lines from 18 to 24 provide the normal use of the Latin Modern Fonts but offer a couple of (commented out) alternatives that may replace the standard Latin Modern fonts: the Times fonts or the Utopia ones. According to my experience, the Latin Modern fonts may appear "too common", but there is a reason; they have optical sizes and are much better suited for technical documents than any other font that does not exhibit this feature. But, of course, this is a question of personal taste.
7. Lines from 26 to 36 configure the necessary fonts for typesetting with LuaLATEX; the font handler fontspec (Robertson, 2019b) is loaded; by default it uses the OpenType version of the Latin Modern fonts; in this example the Libertinus serif, sans serif, and teletype text fonts are loaded; since math is an important part of any technical discipline, the unicode-math package (Robertson, 2019a) is loaded and the Libertinus Math font is selected. A comment specifies that it is possible to specify the math-style=ISO option to the unicode-math package. I strongly recommend to use this option because it fulfils almost all the requirements of the ISO regulation concerning the style of writing prescribed for physics and technology.
8. Line 39 configures the only index that is being produced; the command not only enables collecting the material for the index, but also specifies that an entry for such an index should appear in the table of contents.

9. From line 44 to line 56 there are the conditional settings to configure a customised bibliography; this is just a model, but by experimenting with the example `toptesi-scudo-example.tex` the user can see what it typesets by commenting or uncommenting the class option line 6. Of course every time a style change is done it is necessary to run the typesetter, then the bibliography processor `biber`, than again the typesetter; certainly the user has the necessary experience on these points and it should be superfluous to recommend this procedure.
10. Eventually in line 59, if pdfLATEX is being used, the small package indentfirst (Carlisle, 1995) is called for indenting the first paragraph of each sectional unit; by default such first paragraphs are not indented, while with LuaLATEX they are indented. This little package provides an identical behaviour with both typesetting programs.
11. The last declaration in line 60 specifies that pages should be typeset with a ragged bottom. Of course this is an optional setting; this particular example file contains many large objects, therefore, in order to vertically justify the text block, some glue would be inserted between paragraphs and above and below objects in display; this procedure guaranties that all text blocks have the same height, but that filling white space is very annoying. Personally I prefer a few pages to have uneven bottoms than to have filling white space. Of course, with a different contents, the specification `\raggedbottom` might be superfluous.

The user can examine the source file of `toptesi-scudo-example` and examine what has been omitted from this description. Also the typeset document explains certain details, besides displaying several examples.

The user can copy the `.tex` file to another file while changing its name: and s/he can play experiments by changing some settings, or by testing some other functionalities.

12.2 Splitting the source `.tex` file

The use of commands `\includeonly` and `\include` help maintaining everything connected with single chapters or initial or ending parts of the thesis pretty clear and separate. The preamble of the example file contains this short stretch of code:

```
\includeonly{%
Chapter1/chapter1,%
Chapter2/chapter2,%
Chapter3/chapter3,%
Appendix1/appendix1,%
Appendix2/appendix2,%
References/biblio%
}
```

The various chapters are simple files starting with the `\chapter` command and possibly ending with `\endinput`; they do not contain any preamble and are saved in their own folder; all folders must be subfolders of the one that contains the main file.

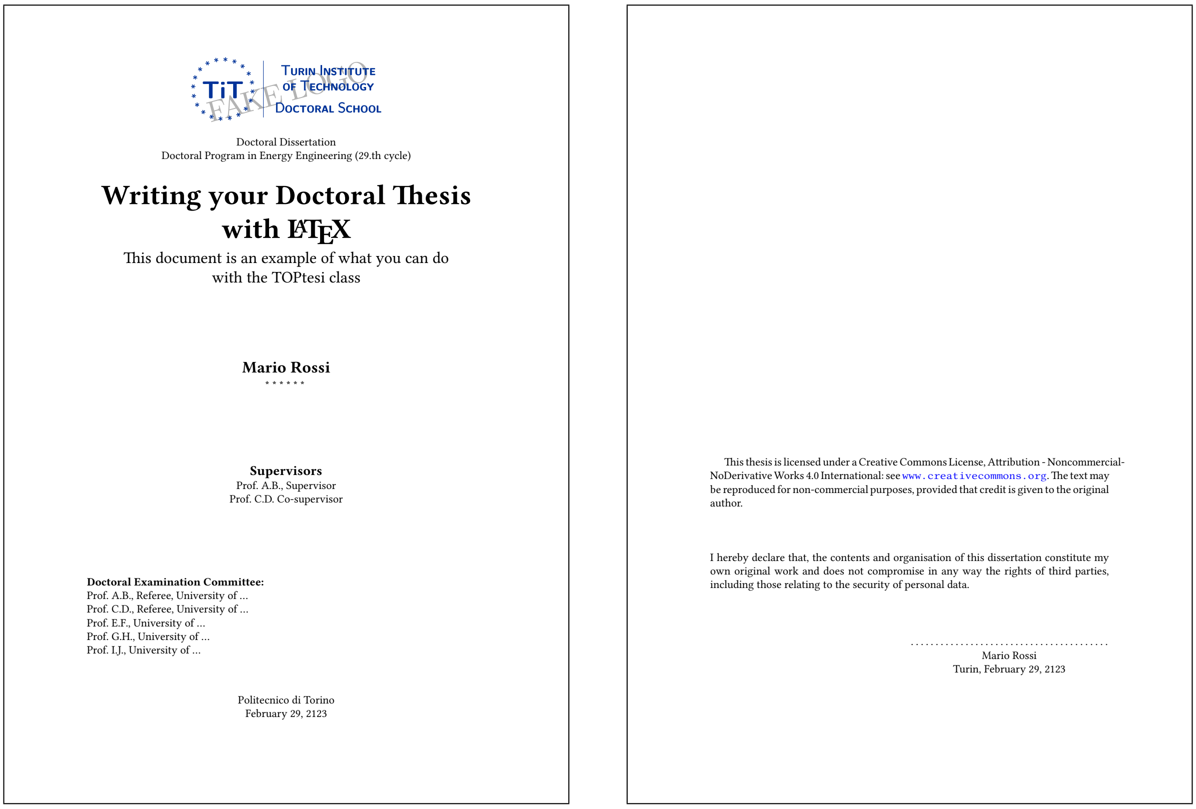
The list of files to be included are written one per line; in this way if one comments all lines except one, it is possible to typeset just that file, with the advantage that all cross reference data of the other files remain available, of course only if the other files have already been typeset. In this way one might compile the first chapter alone; when it is almost complete, its line gets commented out and the next line is uncommented; therefore a new typesetting run is done, and only chapter 2 is typeset and any cross reference to chapter 1 is correctly used. The process proceeds quickly as far as the end of the document, without loosing time in typesetting again and again chapters that have already almost reached their final state. When all chapters are ready, it is time to uncomment all the included file names, in order to fix some residual details, and the final typesetting run creates the whole document.

Of course this trick is applicable to any file to be typeset with any LATEX typesetting engine; here it is recommended because the completion of a doctoral thesis, with its delicate structures and advanced technical language in English (which is the default language for the ScuDo doctoral theses) requires a very attentive reading and correcting of the various drafts; splitting the job into simpler parts eases the task.

12.3 The ScuDo title page and the legal page

Again the example file shows how to typeset the title page and the legal page. The code of the file is highly commented in order to explain variant possibilities. Here I strip the comments out and describe the basic ThesisTitlePage contents.

```
\begin{ThesisTitlePage}
\PhDschoolLogo{TiTDocScCropped}%
             % Fake logo for this example
\ProgramName{Energy Enginering}
\CycleNumber{29.th}
\author{Mario Rossi}
\title{Writing your Doctoral~Thesis\\
  with \LaTeX}
\subtitle{This document is an example
    of what you can do\\with~the~TOPtesi
    class}
\SupervisorNumber{2}
\SupervisorList{%
    Prof.~A.B., Supervisor\\
    Prof.~C.D. Co-supervisor}
\ExaminerList{%
```



TiT Turin Institute of Technology Doctoral School

FAKE LOGO

Doctoral Dissertation
Doctoral Program in Energy Engineering (29.th cycle)

Writing your Doctoral Thesis with LATEX

This document is an example of what you can do with the TOPtesi class

Mario Rossi
* * * * * *

Supervisors
Prof. A.B., Supervisor
Prof. C.D. Co-supervisor

Doctoral Examination Committee:
Prof. A.B., Referee, University of ...
Prof. C.D., Referee, University of ...
Prof. E.F., University of ...
Prof. G.H., University of ...
Prof. I.J., University of ...

Politecnico di Torino
February 29, 2123

This thesis is licensed under a Creative Commons License, Attribution - Noncommercial-NoDerivative Works 4.0 International: see www.creativecommons.org. The text may be reproduced for non-commercial purposes, provided that credit is given to the original author.

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

....
Mario Rossi
Turin, February 29, 2123

Figure 3: The title page and the legal page

```
Prof.~A.B., Referee, University of \dots\\
Prof.~C.D., Referee, University of \dots\\
Prof.~E.F., University of \dots\\
Prof.~G.H., University of \dots\\
Prof.~I.J., University of \dots}
\ExaminationDate{February 29, 2123}
\Disclaimer{%
\noindent I hereby declare that, the
contents and organisation of this
dissertation constitute my own original
work and does not compromise in any way
the rights of third parties, including
those relating to the security of
personal data.
}
\end{ThesisTitlePage}
```

The example makes reference to the PhD School logo; in this case the imported logo is a fake one, because the original ones are downloadable only by the students of the ScuDo doctoral school from a reserved Internet address that I cannot access; the School Registrar enables each student to access the site.

All the personal names of this example are fake ones; even Mario Rossi is the Italian equivalent of the proverbial John Smith in English speaking countries.

As it may be seen in figure 3 the logo is at the top of the page and the other pieces of information are distributed on the page with adequate labels. Such labels might be changed (when allowed), by using special commands that can be found in the English part of the already mentioned documentation file `toptesi-it.pdf`.

In figure 3 there are the statements for the licence related to the contents of the document, and the disclaimer where the signer, who is supposed to sign in original on the dotted line, declares that s/he respected both the intellectual property of others and that s/he did not violate the privacy rights of third persons. This disclaimer has been agreed upon with the Director of the ScuDo doctoral school and cannot be changed, while the “signature” part may be customised.

13 Structuring the thesis

It is not mandatory to segment the whole document in partial files, as already described, but is very useful.

It is very important to divide the inner material in front, main and back matters; each of these divisions may be further divided.

In table 1 it is possible to see the sequence of the major sections, front, main and back matter, and the placement of numbered and unnumbered chapter level segments of a PhD thesis.

Most of the major sectioning with TOPtesi is automatic and does not require intervention from the user. Possibly the user has to avoid using the starred or unstarred `\chapter` command in the front matter, because the first occurrence of this

Table 1: Stucture elements of a PHD thesis

Front matter	
Half title	Generally absent in a PhD thesis.
Title page	As specified by the ScuDo doctoral School.
Legal page	As specified by the ScuDo doctoral School.
Dedication	It is better to avoid dedications in PhD theses.
Acknowledgements	Very seldom necessary. See below
Foreword	Not necessary in a PhD thesis.
Table of contents	It may be followed by the list of figures and the list of tables.
Introduction	A PhD thesis should have an introduction, an unnumbered chapter that may appear in the table of contents. The author describes the structure of the thesis and shortly describes the motivation of performing his/her research in that particular domain; sometimes, and in very special circumstances s/he may acknowledge the support received by other people external to Politecnico di Torino, or external Institutions.
Main matter	
Numbered chapters	A sequence of structured chapters, divided in sections, subsections and other similar hierarchical divisions; generally sections below subsections are not numbered; even if they are, they do not appear in the table of contents.
Numbered appendices	Appendices, if any, appear at the end of the main matter only if they are more than one, and therefore are numbered.
Back matter	
Single appendix	A not numbered single appendix appears at the beginning of the back matter.
Bibliography	One or more bibliographies appear after the possible single appendix; generally there is just one bibliography, but sometimes the references may be arranged in separate lists. Often the search of information useful for the research at the base of the thesis was found in the Web. Each website entry should contain the date of the last visit, otherwise the information is useless.
Glossary	Glossaries, nomenclature lists, acronym lists, and the like are optional.
Index	Optional in a PhD thesis, but not useless; one or more indices are generally present in reference manuals rather than in theses.

command switches from front to main matter settings. Alternative commands for a summary or a dedication or an acknowledgements section are already available in order to avoid an unwanted shift to main matter.

But the problem may arise only with the Introduction; if this consists of a few pages and is not structured, an unnumbered chapter is in order, but if this introduction is long and possibly structured, then it is better that it is a numbered chapter and therefore the normal `\chapter` command is in order.

If a long not numbered and not structured Introduction is necessary the workaround is this:

1. Do not use roman numerals for the folios of the front matter.
2. Use the following code

   ```
   \setcounter{secnumdepth}{-1}
   \chapter{Introduction}
   ⟨Introduction body⟩
   \setcounter{secnumdepth}{2}
   ```

   In facts the level for numbering chapters (see table 2) is lowered to -1; then after the Introduction is finished and before using again the `\chapter` command the level is restored to its original value of 2. It is impossible to use groups to limit the scope of the new setting of the LaTeX counter to the group, because the `\setcounter` command assigns a value to the named counter in a global way.

14 The main matter

The main matter has all sectional units well numbered in a hierarchical way until a level that by default is 2; see table 2; they are listed in the table of contents to level 1. These values may be modified by changing the values contained in a couple of LaTeX counters:

```
\setcounter{secnumdepth}{⟨level⟩}
\setcounter{tocdepth}{⟨level⟩}
```

If the user wants to customise the table of contents in such a way that it contains also the entries for other sectioning commands, s/he can use the command `\setcounter` because LaTeX counters may be assigned values only this way; but remem-

Table 2: Levels of sectional units

Sectional unit	Level
part	-1
chapter	0
section	1
subsection	2
subsubsection	3
paragraph	4
subparagraph	5

ber: the value assignment is global and there is no grouping that may delimit its scope.

Obviously each chapter is long enough to be sectioned in a hierarchical way. Remember: it is meaningless to create a chapter with just one section, as well as to create one subsection within one section, and so on. Once a sectioning level contains any number of sectioning units of higher level, it cannot stop until a new section of the same level is started. More clearly: section 2 of chapter 3 is divided into subsections the first of which needs not be immediately after the title of section 2, but the last subsection of section 2 ends only when section 3 starts.

Any sectional unit contains plain text and may contain some material in display. Let us distinguish floating objects from fixed position objects in display. The former objects float until the typographical rules encoded into the class let them be extracted from their stack and actually inserted in the output pages. The latter are relatively large objects of different nature: they may be titles, formulas, tables, figures, and so on, and they cannot be broken across pages.

14.1 Formulas in display

A simple formula in display occupies a vertical space on the page that is equivalent to at least three lines of text; if the formula contains large operators and fractions it may occupy even more vertical space; if the formula is very long, it may occupy a lot of vertical space that cannot be broken across pages; arrays of formulas occasionally might be broken across pages if suitable commands are used and, certainly, if the array of formulas is not grouped by a large brace.

Let us make an example: everybody knows that the second degree equation[3]

$$x^2 + 2ax + b = 0$$

has its solutions given by

$$x_{1,2} = \begin{cases} -a \pm \sqrt{a^2 - b} & \text{if} \quad a^2 > b \\ -a & \text{if} \quad a^2 = b \\ -a \pm \mathrm{i}\sqrt{b - a^2} & \text{if} \quad a^2 < b \end{cases}$$

It is evident that the three forms of the solution cannot be split across page breaks (nor across columns when typesetting in twocolumn mode).

In any case the toptesi.cls class with option tipotesi=scudo preloads a number of packages; among these it preloads the amsmath (American Mathematical Society, 2018), amsthm (American Mathematical Society, 2017), and, only when typesetting with pdfLATEX, amssymb (American Mathematical Society, 2013); therefore the whole machinery made available by the American Mathematical Society is already available; the user is invited to read the documentation of these packages, because they offer the user a large number of functionalities very useful for technical writings.

In typesetting mathematics the user should pay attention to the ISO rules concerning physics and technology (ISO-31/XI, 1978); s/he should pay a lot of attention to the various fonts that *must* be used for this kind of math. In short terms the ISO rules say the following.

1. Math italics must be used for all scalar variables and physical constants.
2. Serifed upright fonts must be used for function names, mathematical constants, and super- and sub-scripts that represent appositions to the variables.
3. Bold italics are used for matrices and vectors.
4. Bold roman fonts are used for sets, for which the double stroke blackboard bold, if available, may also be used.
5. Upright sans serif fonts are used to label objects in descriptive physical diagrams.
6. Oblique sans serif fonts are not used.
7. Upright bold sans serif fonts are not used.
8. Oblique bold sans serif fonts denote tensors.

Pay attention to Greek letters: by default pdfLATEX uses oblique lower case letters and upright uppercase ones; ISO rules require them all to be oblique when they represent quantities; with LuaLATEX and the option math-style=ISO to unicode-math this situation is corrected; with pdfLATEX one may resort to the pm-isomath package that provides a lot of commands to use the proper family, series and shape for all symbols, not only the Greek ones. But observe the following three different uses of the symbols rendered with a Greek pi: $\mathrm{\pi}$ designates the transcendental math constant $3.141\,592\,653\,589\,793\ldots$; π represents the measurable quantity “flat angle”; $\mathsf{\pi}$ represents the physical particle “pion”; $\boldsymbol{\mathsf{\pi}}$ may represent a tensor. The pm-isomath package is incompatible with

[3] Sometimes the general solutions of the secondo degree equation refer to general coefficients, such as in $ax^2+bx+c=0$. Elementary manipulation of the general equation, brings it to the form used here.

LuaLaTeX and is automatically skipped if the document is typeset with this program. Moreover the commands to address the various fonts properties and the single required glyphs are different from those used with unicode-math with the math-style=ISO option; see the documentation of both packages to find out the differences. See also the documentation of package isomath, (Milde, 2012); pm-isomath was created to override some isomath limitations; in facts the latter package provides full functionality only when using certain math fonts; pm-isomath supposedly works with any font collection, but with some compromises.

Of course there is more than the above short summary, which can be used as a simplified reminder that things are a little more complicated than what we might remember from our early studies.

For physicists and technologists (engineers) the ISO regulations forbid empirical equations between measures, because only the relations between quantities are admitted. Furthermore only the SI units are allowed and any other unit of measure is forbidden; therefore no CGSm, no CGSe, no CGS-Gauss; no British or American units. The use of the siunitx (Smith, 2018) package helps very much to use quantity equations, to typeset tables containing quantities, and so on.

One of the difficult things to remember is the fact that appositions to the quantity symbols must be typeset with roman fonts, while mathematical subscripts must be typeset with italic math fonts: therefore V_max is correct, while V_{max} is wrong; if V represents a voltage, V_i may denote an "input" voltage, while V_i represents the i-th element in an ordered set of voltages.

Package amsmath allows to typeset matrices, equation alignments; long expressions split at proper points in order to fit the measure, and many other features. For matrices see some examples in table 3.

Table 3: The six matrix types that can be created with the various commands made available by the amsmath package. The matrices in this table are built with the environments matrix, pmatrix, bmatrix, Bmatrix, vmatrix, Vmatrix

$$\begin{matrix}1&2&3\\4&5&6\\7&8&9\end{matrix} \qquad \begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix} \qquad \begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$$

$$\begin{Bmatrix}1&2&3\\4&5&6\\7&8&9\end{Bmatrix} \qquad \begin{vmatrix}1&2&3\\4&5&6\\7&8&9\end{vmatrix} \qquad \begin{Vmatrix}1&2&3\\4&5&6\\7&8&9\end{Vmatrix}$$

File `toptesi-scudo-example.tex` contains some more examples. But I think that the best way to become an "equation expert" is to practice with all commands and environments described in the documentation of the amsmath package.

Beware: most office suits and some TeX system installations offer apps or dialog panes called "equation editors". They might appear useful at the very beginning, when the user is not yet comfortable with the myriad symbols and environments offered by the LaTeX kernel and the various extension packages. Such devices seduce the beginners, as well as they are seduced by editors like LyX that promise to display in the editor pane the result of a TeX system typesetting.

The above warning is to stress the point that LaTeX is a particular programming language that allows to define new functions and functionalities; those ready to use graphic user interfaces are prebuilt to do certain things while speeding up the graphic visualisation of the result. Equation editors and editors like LyX cannot do more than what they are programmed to do; actually modern versions of LyX have functions capable to translate TeX programming into LyX language; therefore if you want to use a specific LaTeX extension package, you can feed it to LyX so that it may translate the package code to its own language and save the result into a file that can be used in future instances. Time consuming, but LyX effective. But you cannot define your own commands to ease your own work, unless you prepare an extension package and submit it to the above mentioned LyX preprocessing. Why then using an editor that requires such preprocessing if any other editor works directly on your source file with no restrictions? Just to see in the editor pane something that resembles to what you hopefully would like to obtain from TeX? By experience I can say that in a long run it amounts in a waste of time, and it is not worth the amount of time you spend in preprocessing.

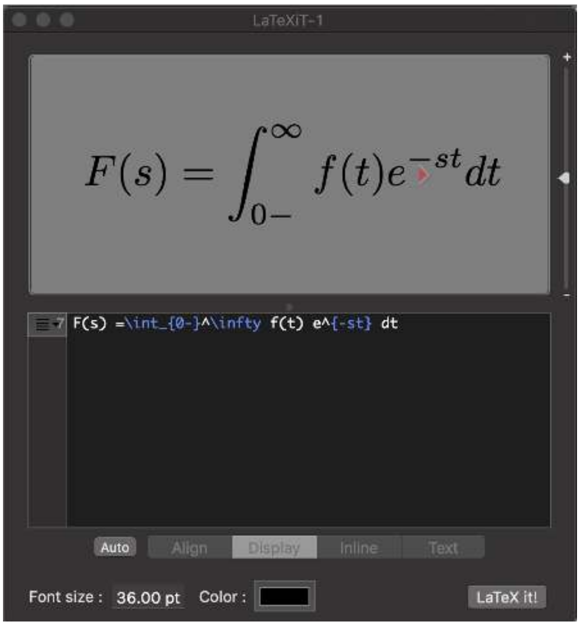
An example: figure 4 displays the editing pane of `LaTeXiT.app`, the equation editor installed on a Mac, when MacTeX, i.e. TeX Live for Mac, is installed.

As you see in the screenshot shown in figure 4 in the lower pane where you have to write LaTeX code; when you are finished, you click the bottom right button "LaTeXiT", and the app shows the result in the upper pane. Isn't that nice? Not quite: it recognises only the standard LaTeX commands, not the extensions of any package, and in particular the settings for fulfilling the ISO regulations that require a roman 'e' and a roman 'd' and suitable spacing before the integrator.

Even if the `LaTeXiT.app` is a fine piece of software, and the other equation editors generally behave more or less in the same way, don't use equation editors! You get stuck within their limits.

14.2 Figures

We already said that LaTeX can include graphic files in the formats PDF, EPS, and MPS (possibly vectorial), and JPG (lossy compressed bitmapped), and PNG (lossless compressed bitmapped). LaTeX



```
F(s) =\int_{0-}^\infty f(t) e^{-st} dt
```

Font size : 36.00 pt Color :

Figure 4: The graphic interface of the `LaTeXit.app` for Mac

can include drawings composed with its internal environments, and the various packages that allow advanced programmed drawing. There are also external programs that can produce output in the form of `.tex` files. At least one of these programs, `asymptote`, is installed together with TEX Live. This free software program is very useful for technical drawings; see for example (De Marco, 2009).

The user can install for example `gnuplot`, a mathematical drawing software: you enter a text file containing the necessary information about the functions to be drawn and the settings for the diagram, and the instructions for the output. With suitable output settings, the result is a `.tex` file that can be directly input into the user's thesis; see (Miklavec, 2013) to examine these functionalities; but you have to download the `gnuplot` program from its site `https://sourceforge.net/projects/gnuplot/files/gnuplot/`; there are versions for the three main platforms Windows, Mac, and Linux; but there are also the source files, so that it is possible to customise the executable program to the specific needs of the user's computer and operating system.

Another external free software program is `Inkscape`; it provides a graphical interface to draw almost anything; a nice part is that the labels that identify the various parts of the drawing are output as a separate file that superimposes the labels (typeset with the current fonts in the TEX output) on the vectorial PDF graphic file it produces. This property is highly desirable, and it is not very common that external programs can guarantee such functionality.

In facts some diagrams may be created from numerical tables set up with, say, Microsoft `Excel` and exported in PDF format. But as anybody can experience, the fonts used to label the axes and other similar text labels are clearly set with fonts that have nothing to do with those used to typeset the user's thesis.

The native LATEX extension packages, such as the extended picture environment, the PSTricks bundle, the TikZ bundle, the pgfplots bundle, all create fine drawings while using the current fonts in the source file. I would suggest to keep in mind such extension packages, because the graphic output is certainly very professional and typographically correct.

Some years ago a frequent user of the Italian TUG forum, nicknamed Liverpool, opened a thread dealing with tracing of lossodromic and orthodromic routes on a 3D sphere projected on the plane together with meridians and parallels, and tracing those arcs in the background of the sphere with thinner lines than those in the foreground. I was very happy to participate in this program and eventually I wrote the `.dtx` file that documented the PGF/Ti*k*Z library used to trace some routes in order to see the different paths along the orthodromic vs. the lossodromic route[4] joining two given points. The library eventually contained also the specific commands to draw the two routes on a gnomonic projection and on a Mercator map (cylindrical projection). Liverpool eventually sent me back the `.dtx` file that I correctly signed only with his name; up to now this library is unpublished. I think it is instructive to see at least the results we reached.

Figure 5 displays a couple of examples; on the left the routes connecting New York and Moscow are displayed while on the right the corresponding routes joining New York with Bangkok. The code to draw one of the two maps is the following.

```
\documentclass{standalone}
\usepackage{pgfplots}
\pgfplotsset{compat=1.11}
\usetikzlibrary{quotes, rotte}

\begin{document}
\begin{tikzpicture}
\begin{axis} [x={(-0.866cm,-0.5cm)},
              y={(0.866cm,-0.5cm)},
              z={(0,1cm)},
              anchor=origin, at={(0,0)},
              disabledatascaling,
              hide axis]
% Draw the globe
\addplot3 [surf, z buffer=sort,
```

[4] Just to remember: the orthodromic route is the shortest path on a spherical surface that joins two given points; the lossodromic route is the path joining the given points such that the compass bearing remains constant; the orthodromic route, therefore, is an arc of a great circle, while the lossodromic one, unless the given points lay on a parallel, is an arc of a sort of spiral that winds on the spherical surface from one pole to the other.

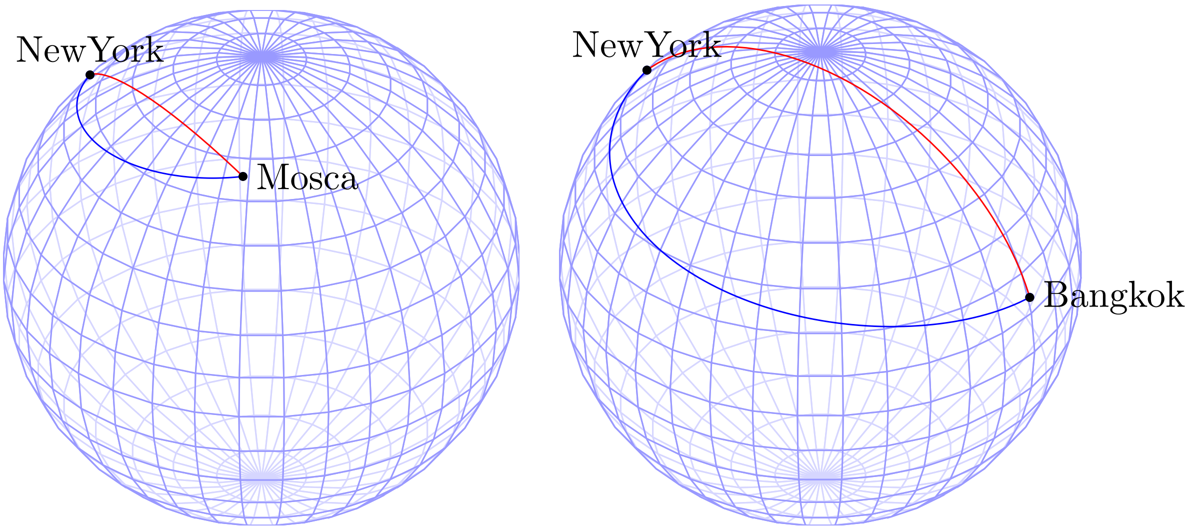


FIGURE 5: Two examples of routes joining two cities on the globe that are relatively close (New York and Moscow) or distant from one another (New York and Bangkok). The red route is the orthodromic one and the blue one is the lossodromic one.

```
           fill opacity=0.6, white,
           faceted color=blue!40,
           samples=19, samples y=37,
           variable=\u, variable y=\v,
           domain=0:180, y domain=0:360]
           ({2*cos(u)*sin(v)},
           {2*sin(u)*sin(v)}, {2*cos(v)});
% Draw orthodromic route
\pic [partenza={(40.744,-73.982)},
          arrivo={(13.725,100.51)},
          raggio=2cm, red] {ortodromia};
% Draw lossodromic route
\pic [partenza={(40.744,-73.982)},
         arrivo={(13.725,100.51)},
         raggio=2, smooth, samples=50,
         mark=*, mark size=1pt,
         blue] {lossodromia};
% Label departure point
\pic [posizione={(40.744,-73.982)},
        raggio=2, mark=*, mark size=1pt,
        "NewYork", above] {coordinata=A};
% Label arrival point
\pic [posizione={(13.725,100.51)},
        raggio=2, mark=*, mark size=1pt,
        "Bangkok", right] {coordinata=B};
\end{axis}
\end{tikzpicture}
\end{document}
```

Some picture qualifiers are in Italian, because Liverpool and I did not work out this TikZ library for international use; here I just changed the comment lines to English language. The interesting point about these drawings is that geographical coordinates of the departure and arrival points are just given in (fractional) degrees of latitude and longitude on the globe. This example shows quite well the power of the graphic packages that are part of any updated and complete TEX system installation. The code is not terribly complicated; the difficult part, if nothing else is already available, is to write the macros shown above and that may be saved in a TikZ library.

TABLE 4: The Smith family

The Smith family			
Name	Role	Age	Activity
John	father	47	employee
Mary	mother	44	primary school teacher
Johanne	daughter	14	junior high student
Peter	son	8	primary school pupil

14.3 Tables

Tabular typesetting is possibly the most difficult task in typography. When typographers were using metal types, they would charge extra money for books and other printed material that contained tabular material. Of course with LATEX everything is much easier, but creating professional tables is still something that is out of our common experience; moreover, since most people create tables using word processors, and since this kind of software has limited performances, most of the time the tables we happen to read are typeset in a manner that is far from professional.

Of course a PhD thesis should have professional tables. Some tables are already shown in the mentioned `toptesi-scudo-example.pdf` where the same table is typeset in a non professional way, in a better way and in a professional way.

But here we should go into the details.

In table 4 you can see a small table typeset with the rules of the best typographical practice.

Notice the details.

1. The table does not contain any vertical rule.

2. The three horizontal rules have different meanings; the first and the last are thicker and delimit the table. The thinner middle rule separates the column headings from the other column cells. No other rules are necessary and even the first and last ones may be omitted. There are very rare occasions when a middle rule or a partial middle rule (spanning just a subset of columns) might add something to the table "meaning". LATEX does not directly provide any means to fix the rule widths, but the (almost compulsory) booktabs package (Fear, 2016) comes to rescue; in practice these package facilities should be used for any table.
3. There is no need to emphasise the table headers with boldface or italic fonts, but it is not forbidden.
4. The cells contents are left or right justified within their cells; they might be also centred or form narrow paragraphs; in this last case it is better that the paragraph is typeset ragged right; LATEX provides these facilities, but the array package helps very much with other paragraph-like cell contents and with very useful functionalities to customise whole columns or single cells.
5. Any table cell may contain another full table, i.e. the tabular environment may be nested.
6. In most countries, Italy included, the table caption is set over the tabular material as in table 4. In English speaking countries tables have their caption under the tabular material. There is no best practice, in the sense that both placements are correct in the proper country.
7. The example table natural width is too large compared to the column width, but not so large to suggest a top page centred full width float position. This is a common situation even if the document is being typeset in onecolumn mode. In this case the solution that I chose consists in reducing the font size; this is simple if the used fonts are continuously scalable, or better, if they are piecewise continuously scalable because they have available optical sizes. In this paper, typeset with the Latin Modern font collection that has optical sizes, the chosen solution appears to be the most comfortable one. Characters of size 8.5 pt are too small to be read? Yes, may be; in this case another solution might be to reduce the inter-column whitespace width; by default it is 12 pt wide, approximately 4 mm; there is enough space to reduce the whole table without reducing the font size too much. See then table 5 where the font size is 9.5 pt and the inter-column width is just 6 pt.

Table 5: The Smith family

The Smith family			
Name	Role	Age	Activity
John	father	47	employee
Mary	mother	44	primary school teacher
Johanne	daughter	14	junior high student
Peter	son	8	primary school pupil

The above small list describes the best practice, but it also underlines the big or small problems that come up with tables. By typesetting in onecolumn mode, as the PhD ScuDo style requires, some of the described problems may vanish or are reduced. One suggestion I can give for tables that are too wide, is to use the widetable (Beccari, 2018) package with its widetable environment that computes the necessary inter-column width in order to fit into a specified width; of course, should it compute a negative inter-column width, it issues a warning and typesets the tabular material with the default value.

In other circumstances the solutions might be of different nature; one is to use the X column descriptors that produce paragraph-like cell contents, but their width is automatically computed to fit the specified table width.

Another solution is to typeset a definitely too large table in a sideways mode; the best practice requires to rotate the sideways material 90° counterclockwise independently of the fact that this material falls on an even or odd page. Several packages available for such rotations perform it in such a way that the caption base is also facing the outer trim margin. This is why I am not naming any particular package.

All "regular" tables are not broken across page breaks; since they are large objects, they need to be floated, so that LATEX can find the best place to output them. LATEX must fulfil certain conditions to output all kind of floats: how tall are they? how many at maximum may be in the same page? in which position on the page: top, bottom or within the text? how much residual space remains on the page for regular text? when in twocolumn mode should the float be in some position of a column or should it span the whole text block width? Actually this may be decided by the user, but again, depending on the type, the specific constraints similar to those for onecolumn mode remain valid. Such constraints are somewhat released when the float stack still contains some floats, but the chapter is finished and they must be all output before a new chapter starts; or when the end of the document is reached.

But if a table is definitely too long it may be typeset on several consecutive pages; the environments longtable and supertabular can do the job; I feel more comfortable with longtable, but it is just a question of personal taste: their performances are almost the same. Provided they are not wider than the text block height, there are also packages to typeset long tables in a sideways manner, continuing to use the same rotation, page after page, as far as the end of the table. I do not like such sideways long tables, but I understand that in certain circumstances it is impossible to avoid them. I think it is better to plan their construction in a smarter way. Since automatic procedures cannot be smarter than a human, humans should not rely on the limited intelligence of those procedures.

15 Bibliography

The bibliography is an important part of a doctoral thesis; it is not meant to document every piece of information that appears in the thesis. It should list the documents effectively read, the references that mostly describe the state of the art, not only in the discipline where the thesis may be classified, but also in side disciplines that were used during the development of the thesis.

A bibliography should not emphasise the visited Internet sites, even if some information or some data were taken from those sites; the explanation is simple; even if they do not transmit superficial or even fake data, they are not reliable; today they are accessible; tomorrow they might have disappeared from the Web. If the thesis is read by someone else in, say, ten years from now, it is very likely that more than a half of those sites will not exist anymore.

This is a common experience, therefore I avoid citing material that is not recorded in a stable way; in this paper I cite TEX related documentation because I assume that if you, the reader, are using the TEX system, you have the references that I cite on board of your computer.

15.1 The bibliographic database

The first step to create a bibliography is to create a database where each record describes everything connected with each reference in a formal way. There are at least two programs that ease this task.

1. The external program `JabRef` is a program that runs in a Java virtual machine; it is suitable for any operating system provided it has the Java bundle installed; `JabRef` is fully compatible with the TEX system. The record names are sort of standard and the descriptors of the various fields of each reference are consistent and standard; if one uses non standard descriptors, they are simply ignored together with the description they address. The program has its own graphical interface.
2. The external program `BibDesk` is only for Mac platforms and is already available when TEX Live is installed with MacTEX. This program has its own graphical interface.
3. Actually there is a hard way to create a bibliographic database: it consists in using the same text editor that is being used to handle the `.tex` files. The user can use it to create a `.bib` file containing the textual material of each reference record; of course s/he should have a clear understanding of the mandatory and the auxiliary information that any type of reference requires. The first two methods are much superior because those programs can fill in the mandatory descriptors for every document type and let you add any other descriptor or field that you find useful, even if it will not migrate to the final typeset bibliography.
4. It is strongly recommended to name the database with the same name as the main thesis file, but, of course, with the mandatory extension `.bib`. It is possible to enter a comma separated list of database files into the argument of `\addbibresource`. A thesis should not contain hundreds of references, so that it is not necessary to split a huge single database in a number of smaller chunks; sometimes it may be useful to keep the list of URLs, the books, the articles, the internal reports, and so on in separate databases. In this case extreme care should be paid to the fact that the citation keys must be unique among the whole set of databases.

An example of bibliographic database is the following.

```
@manual{man:ReferenceManual,
        Author = {Claudio Beccari},
        Address =
        {\url{http://www.guitex.org/home/images
            /doc/GuideGuIT/}},
        Organization = {{\GuIT}},
        Title = {{Il \LaTeX\ Reference Manual
                  commentato}},
        Year = {2017}}

@misc{misc:toptesi,
        Author = {Claudio Beccari},
        Howpublished = {{PDF document}},
        Note = {in
         {\url{$TEXMF/doc/latex/toptesi/
                 toptesi.pdf}},
         \url{$TEXMF/doc/latex/toptesi/
                toptesi-it.pdf}},
        Title = {{La classe \pack{TOPtesi}}},
        Year = {2019}}

@book{book:Bringhurst,
        Address = {{Vancouver, BC}},
        Author = {Robert Bringhurst},
        Publisher = {Hartley \& Marks},
        Title = {The elements of typographic
                  style},
        Year = {2004}}

@article{art:Caignaert,
        Author = {Christophe Caignaert},
        Journal = {\textsl{TUGboat}},
        Number = {3},
        Pages = {161-174},
        Title = {A story of \textit{kpfonts}},
        Volume = {31},
        Year = {2010}}

@manual{man:ShortMathGuide,
        Address = {Providence, Rhode Island},
        Author = {Michael Downes},
        Edition = {1.09},
        Note =
        {In \url{ftp://ftp.ams.org/pub/tex/
           doc/amsmath/short-math-guide.pdf}},
        Organization = {American Mathematical
                        Society},
        Title = {Short math guide},
        Year = {2002}}


@manual{man:Flynn,
        Author = {Peter Flynn},
        Note = {In {\url{CTAN/tex-archive/info/
               beginlatex/beginlatex-3.6.pdf}}},
        Title = {Formatting information ---
                 A beginner's introduction
                 to typesetting with {\LaTeX}}}

@manual{man:xetex-companion,
        Author = {Michel Goossens},
        Note = {in \url{http://xml.web.cern.ch/
               XML/lgc2/xetexmain.pdf}},
        Organization = {{\LaTeX\, Team}},
        Title = {The {\XeTeX} Companion --
        {\TeX\ meets OpenType and Unicode}},
        Year = {2011}}
```

As it can be seen, every record starts with an "at sign" (@), followed by the name of a type of reference; most of these names are full words, but some are simple abbreviations, such as misc (miscellaneous) to be used when the reference is difficult to be classified.

Each record content is enclosed within balanced braces; it is divided in fields and each field has a name followed by the "equals sign", followed by a balanced braces pair containing the field contents; the only exception is the first field that contains the citation key without any name; each field content is actually separated from the next one with a comma. Except for the Author field, all other fields, if they were not enclosed within balanced braces, are transformed to lowercase except for the first word initial letter. This is why it is better to always enclose the field contents within braces. This lowercasing would take place also with LATEX macros; therefore during typesetting those lowercased macros become typesetting errors of the type "Undefined control sequence"; tricky error, because when the source .bib file is examined no errors are found. But if the processed bibliography file with extension .bbl is examined the error is immediately spotted.

There are at least two ways to typeset a bibliography from a bibliographic database that depend on the bibliography external processor bibtex or biber, and on the type of bibliography style chosen.

The ScuDo doctoral thesis is typeset trough the use of the biber processor, therefore here nothing is said concerning the use of bibtex.

The toptesi-scudo.sty module presets the bibliography processing through the biblatex package (Lehman, 2018) and its options in order to get an alphabetically ordered bibliography on the basis of the author's surname, but with each reference identified by a number. The setting is as this:

```
1 \unless\ifmybibstyle
2   \usepackage[autostyle]{csquotes}
3   \usepackage[backend=biber,
4               style=numeric-comp,
5               citestyle=numeric,
6               sorting=nty,
7               natbib]{biblatex}
8   \addbibresource{\jobname.bib}
9 \fi
```

The test starting on line 1 and completed on line 9 preloads the necessary packages and sets the options to describe the bibliography style only if the class option mybibliography was *not* specified by the user. If it was, nothing is done and the user becomes totally responsible to select the procedure and the style s/he prefers *if and only if s/he got an authorisation from his/her supervisor.* This is because the style of the ScuDo doctoral theses should be conformant to a model that lets anyone recognise at first sight the school theses.

On line 2 the csquotes package (Lehman and Wright, 2018) is loaded; this allows to adapt the quotation marks to the style used in the country where a specific language is used. Actually the records of the bibliography database might contain a field that qualifies the reference as one to be written in a certain language; this is useful also for the correct word hyphenation at line breaks.

Then from lines 3 to 7 the biblatex is loaded with the necessary options; specifically:

backend=biber specifies that the biber processor is used to extract and format the various references.

style=numeric-comp specifies that the numeric identification of every reference is used.

citestyle=numeric the same style is specified for citations.

sorting=nty specifies that the sorting is principally based on author names; for equal names, sorting is based on titles; for equal titles, sorting is based on year of publication.

natbib allows to use the same citation schemes established by the natbib package (Daly, 2010).

Line 8 specifies that the bibliographic database file is named \jobname.bib. In his/her thesis the candidate may specify the names of other bibliographic databases with similar commands; the bibliography processor will examine the whole set in order to extract and format the various entries. The important point is that entries in all databases have a unique citation key.

With the default settings the bibliography turns out as in figure 6, while with the settings used in the sample file `toptesi-scudo=example.tex` the bibliography turns out as in figure 7.

All this implies that the `biber` program must be run after at least one compilation of the thesis and after any modification of the databases; but it is not necessary to run it again and again after each compilation of the thesis.

16 Nomenclature

Personally I find a glossary or a nomenclature list, a list of symbols, a list of acronyms, whatever you want to call them, useful just in certain circumstances. Of course a purist may distinguish those lists for many details. But they have something in common. All of them are lists of lexemes, and of each one they explain what they mean or represent.

It is possible that in a doctoral thesis some words appear to be used in a special or restricted way and are not commonly known even to scholars in the disciplinary field. I assume that such terms are very rare and that the examining committee is made up of experts. For the members of this committee a nomenclature list should be superfluous.

Nevertheless it is possible that in the future the thesis is read by people that are not so expert. For these people a nomenclature list might be very useful.

The tipotesi=scudo already loads the nomencl package; and already defines some categories of names to be described. The example file `toptesi-scudo-example.tex` makes some examples of nomenclature entries (with silly definitions, just to show how to use them); but the real bonus is that this TOPtesi module already contains the shell-escape commands necessary to typeset the nomenclature list just in one run. In facts in a "normal" situation the user should enter with similar commands into the main or one of the secondary files the material that forms the nomenclature list; s/he should compile the thesis, then s/he should open a terminal or command prompt, and should enter the necessary operating system command to run the external program `makeindex` that processes the TEX material that the typesetting program already transferred to a raw nomenclature file; the result of this external processing is a definitive `.nls` file, that in the next typesetting run will become the typeset nomenclature. It is more complicated to describe it than to do it; but even so, everything is already hardcoded into the ScuDo module, so that the user should not care about anything related to the nomenclature typesetting.

17 Index

The implemented process to create the nomenclature is similar to the one used for creating an index, but in this case the whole machinery is already included into the imakeidx package that is preloaded by the ScuDo module.

With indices the problem is more complicated than for nomenclatures, because the user might desire to make more than one index at the same time.

Whatever is listed in an index is generally followed not by a description as in a nomenclature, but by the list of the pages in the thesis where something interesting is said concerning each lemma. The page numbers might be typeset with different fonts as it is done in the TEX book; where a boldface page number is used to point to the page where there is the lemma definition, a normal upright page number is where the lemma is used in a significant way; an italic page number may represent the page where the lemma is used in an application, and so on.

During the typesetting task the material to appear in the index or indices is collected and written in one or more raw `.idx` files; before the typesetting task is terminated the written files are transmitted through operating system commands to the external processing program that produces one or more ordered index files; then, always before terminating the typesetting task the processed files are input into the main typesetting flow and the index or indices are typeset. The whole procedure is very handy and I do not use anything else to produce the indices I need. And I think that this way of processing indices and nomenclatures is the most useful one.

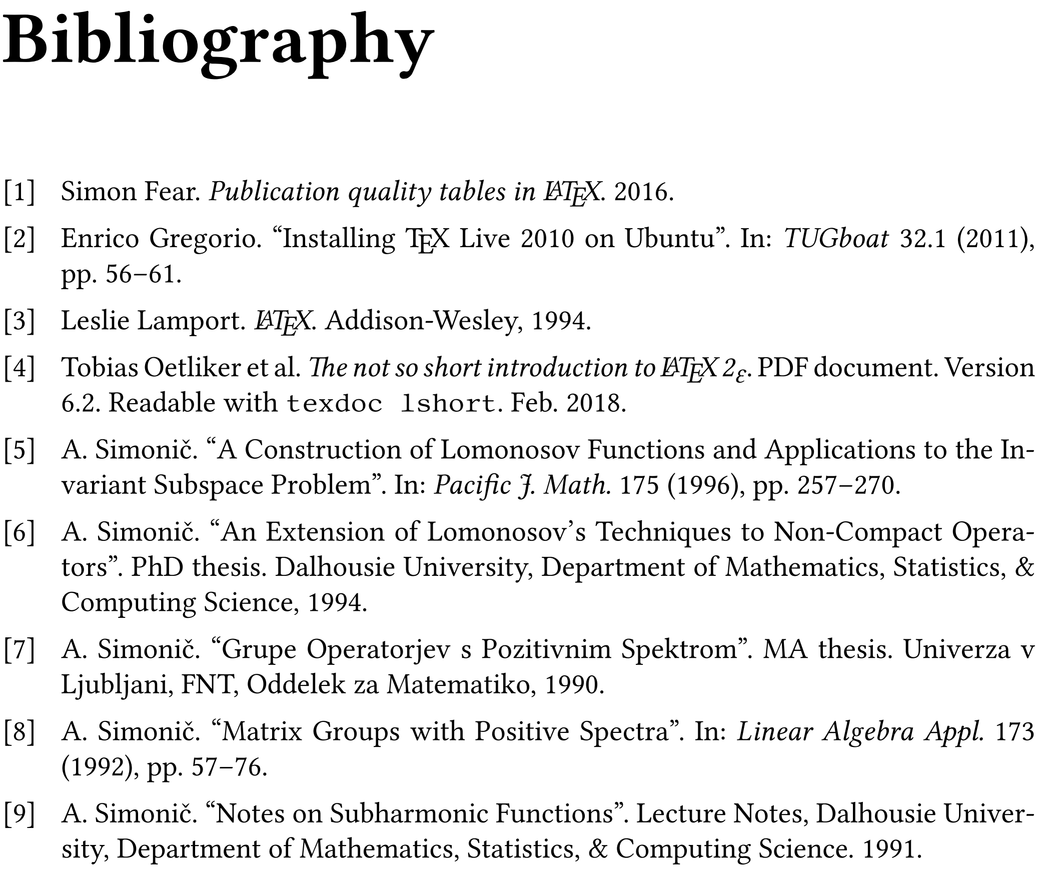
It is assumed, of course, that nomenclature and indices appear at the very end of the typeset thesis, therefore after the appendices and the bibliography.

17.1 Configuring `\makeindex`

The normal use of the indexing facilities require that the preamble of the document contains the `\makeindex` declaration. With package imakeidx this declaration may be customised:

```
\makeindex[%
  ⟨index name⟩,
  ⟨index title⟩,
  ⟨other options⟩]
```

The ⟨*index name*⟩ is a symbolic name to distinguish which index should be activated with the specified options; if this option is omitted, the settings apply to the default index. This optional ⟨*index name*⟩ is also used to configure the `\index` command to send its argument to the specified index.



Bibliography

[1] Simon Fear. *Publication quality tables in LaTeX*. 2016.

[2] Enrico Gregorio. "Installing TeX Live 2010 on Ubuntu". In: *TUGboat* 32.1 (2011), pp. 56–61.

[3] Leslie Lamport. *LaTeX*. Addison-Wesley, 1994.

[4] Tobias Oetliker et al. *The not so short introduction to LaTeX 2$_\varepsilon$*. PDF document. Version 6.2. Readable with `texdoc lshort`. Feb. 2018.

[5] A. Simonič. "A Construction of Lomonosov Functions and Applications to the Invariant Subspace Problem". In: *Pacific J. Math.* 175 (1996), pp. 257–270.

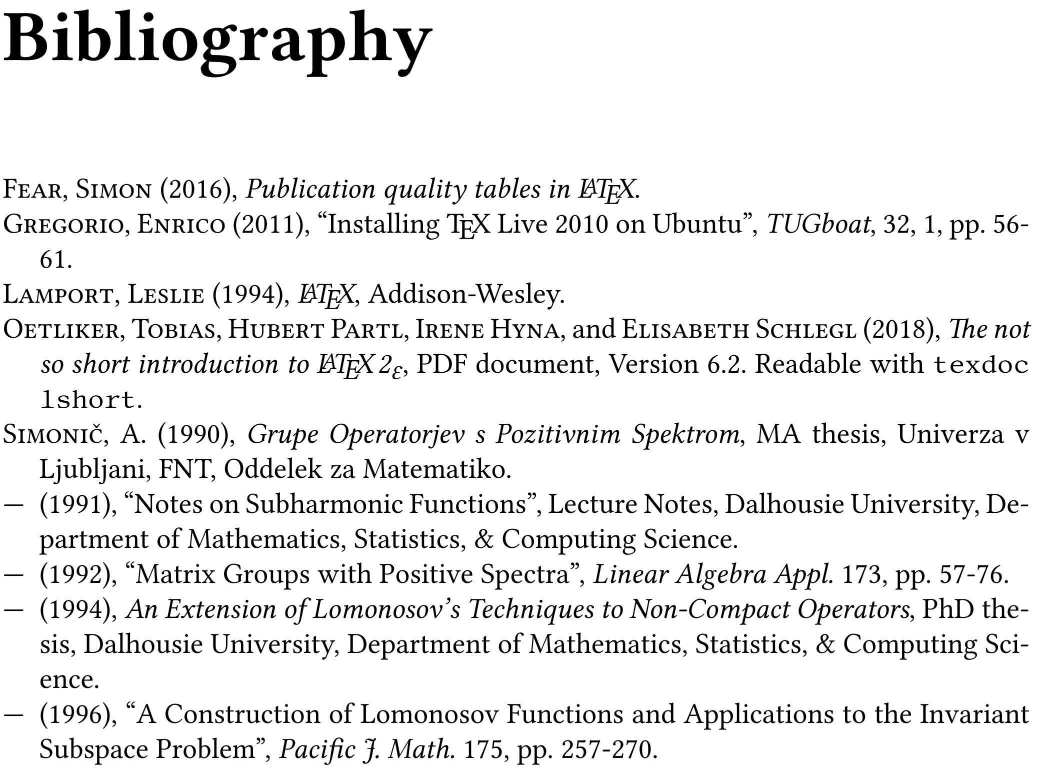
[6] A. Simonič. "An Extension of Lomonosov's Techniques to Non-Compact Operators". PhD thesis. Dalhousie University, Department of Mathematics, Statistics, & Computing Science, 1994.

[7] A. Simonič. "Grupe Operatorjev s Pozitivnim Spektrom". MA thesis. Univerza v Ljubljani, FNT, Oddelek za Matematiko, 1990.

[8] A. Simonič. "Matrix Groups with Positive Spectra". In: *Linear Algebra Appl.* 173 (1992), pp. 57–76.

[9] A. Simonič. "Notes on Subharmonic Functions". Lecture Notes, Dalhousie University, Department of Mathematics, Statistics, & Computing Science. 1991.

Figure 6: Bibliography with numerical labels



Bibliography

Fear, Simon (2016), *Publication quality tables in LaTeX.*

Gregorio, Enrico (2011), "Installing TeX Live 2010 on Ubuntu", *TUGboat*, 32, 1, pp. 56-61.

Lamport, Leslie (1994), *LaTeX*, Addison-Wesley.

Oetliker, Tobias, Hubert Partl, Irene Hyna, and Elisabeth Schlegl (2018), *The not so short introduction to LaTeX 2$_\varepsilon$*, PDF document, Version 6.2. Readable with `texdoc lshort`.

Simonič, A. (1990), *Grupe Operatorjev s Pozitivnim Spektrom*, MA thesis, Univerza v Ljubljani, FNT, Oddelek za Matematiko.

— (1991), "Notes on Subharmonic Functions", Lecture Notes, Dalhousie University, Department of Mathematics, Statistics, & Computing Science.

— (1992), "Matrix Groups with Positive Spectra", *Linear Algebra Appl.* 173, pp. 57-76.

— (1994), *An Extension of Lomonosov's Techniques to Non-Compact Operators*, PhD thesis, Dalhousie University, Department of Mathematics, Statistics, & Computing Science.

— (1996), "A Construction of Lomonosov Functions and Applications to the Invariant Subspace Problem", *Pacific J. Math.* 175, pp. 257-270.

Figure 7: Author-year bibliography

Again the ⟨*index title*⟩ is useful to set a title to each of several index files and to use it in the printed indices. Of course, even if just one index is created, it is possible to give it a title different from the default one.

The ⟨*other options*⟩ refer to other details of each index: for example it is possible to typeset an index in a number of columns different from the twocolumn default mode. The user is invited to read the documentation of the imakeidx package documentation (Gregorio, 2016).

17.2 Entering data to an index

Entering each entry to every index is done through the `\index` macro. This is a very particular macro: first, it is disabled when `\makeindex` declaration has not been specified; second, the macro argument has a very special syntax, because it is fed to the external program `makeindex` that requires that syntax. The original documentation by Leslie Lamport Lamport (1987) dates back to 1987, but with minor variants is still valid. A general document on making indices (Chen and Harrison, 2014) is more recent because it is dated 2014, but still it does not mention the enhancements provided by package imakeidx.

The general syntax for entering index data is the following (to be input in just one line):

⟨*first level*⟩`!`⟨*second level*⟩`!`⟨*sort entry*⟩
`@`⟨*typeset entry*⟩`|`⟨*address*⟩

Where ⟨*sort entry*⟩ is the string used to sort the index, while ⟨*typeset entry*⟩ is what appears in the index; for example, the user wants to enter the word “Transistor” in boldface, but it does not want to sort it among the capitalised entries: then ⟨*sort entry*⟩ will be “transistor” and the ⟨*typeset entry*⟩ will be “`\textbf{Transistor}`”. Moreover ⟨*address*⟩ is the way to typeset the page number, or the string to use so as to point to another entry. The ⟨*first level*⟩ entry and ⟨*second level*⟩ entry are used only when the ⟨*typeset entry*⟩ should appear hierarchically under other entry levels.

It is convenient to define handy macros that use the correct syntax; for example in this paper I use the command `\pack` to enter package names in upright sans serif font: if I wanted to create an index containing the package names grouped under a first level entry “packages”, in the preamble I would define the following code (that requires the use of the xparse package, The LaTeX 3 Project Team (2018)):

```
\newcommand\packagestyle[1]{\textsf{#1}}
\NewDocumentCommand\pack{s m}{%
\packagestyle{#2}\IfBooleanTF{#1}{}%
 {\index{packages!#1@\packagestyle{#2}}}%
}
```

The first definition establishes how I want to typeset package names; the second definition (that requires the `xparse` facilities) defines a command `\pack` that accepts an optional star and a mandatory name of a package. If the first optional argument is a star, the command just typesets the package name, otherwise, besides typesetting the package name in the document, it sends the indexing information to the output `idx` file; the external program `makeindex` (automatically invoked by imakeidx) creates an entry "packages" and a subentry with the name of the package typeset with the proper font. Therefore I have available two ways of using the command `\pack`:

- `\pack*{`⟨*package name*⟩`}` to simply typeset the ⟨*package name*⟩ in the document; and
- `\pack{`⟨*package name*⟩`}` to typeset the ⟨*package name*⟩ in the document and simultaneously send the subentry ⟨*package name*⟩ to the index.

If the user wishes to create more than one index, in the preamble of the document s/he should customise more than one `\makeindex` declaration, for example

```
\makeindex[intoc]
\makeindex[name=places,
           title=List of places,
           intoc, columns=1]
```

By so doing the user sets two indices, the first one is the normal one with the default name and it is listed in the table of contents. The second, named `places`, has the title “List of places”; it is typeset in onecolumn mode; it is listed in the table of contents.

In the body of the document, the user introduces such commands as in this text example:

```
Albert Einstein was born in
Ulm\index[places]{Germany!Ulm@Ulm} and
while he was working in
Bern\index[places]{Switzerland!Bern@Bern}
started working on the theory of
restricted relativity%
\index{relativity!restricted@restricted}.
```

The mechanism is pretty simple to be used; and it becomes simpler if the user defines suitable macros to enter the necessary information just once; the `xparse` package has facilities to define macros that accept almost any kind of mandatory and optional arguments. The difficult part of making indices remains the author choice of which lemma instances s/he wants to mark with the `\index` command.

18 Archivable format

Politecnico di Torino requests a copy of each thesis to be submitted to the registrar when each student submits the formal application in order to defend his/her thesis. These copies must be "dematerialised", i.e. in PDF format, provided that this PDF file fulfils the regulations issued by the International Standards Organisation (ISO) for long term archivability. Such ISO regulations were first issued in 2005, and in the following years they received further additions. The documentation of the pdfx package, (C.V. Radhakrishnan *et al.*, 2018), describes the various levels of ISO conformity and describes how to obtain conformant PDF files by using the package.

The LaTeX based typesetting engines that are part of the TeX system up to now cannot produce files with tagged contents, named *Tagged PDF* files. This means that today such engines can produce PDF files that are conformant only to the PDF/A-1b regulations. In another paper Ulrike Fisher is going to describe the progress in modifying these engines in order to produce tagged PDF files, and therefore files that can be conformant with the more stringent regulation PDF/A-1a and also to other more recent standards.

The constraints a PDF/A-1b conformant document must fulfil deal with some delicate points.

PDF level Even the PDF language underwent to several upgrades so that the ISO established for PDF/A-1b files that the PDF language level should be exactly 1.4, no less, no more. More recent ISO standards allow higher language levels.

Fonts The fonts used in the document must be vectorial; bitmapped fonts are absolutely forbidden. Glyphs with zero width are forbidden. Unfortunately some standard TeX math fonts do not fulfil such constraint. The TOPtesi bundle contains a patch to this problem, but actually there is no need to do any patch when the typesetting engine is LuaLaTeX and Unicode math fonts are used.

Encoding UNICODE and certain TrueType fonts are accepted; Type 1 fonts are accepted only if the file with the correspondence of their glyph addresses with UNICODE is included. For this reason it is better to avoid Type 1 fonts and therefore it is necessary to typeset the PDF document with LuaLaTeX. It would be possible to use also XeLaTeX, but special postprocessing would be required.

Colors Colors should be only RGB (and/or grayscale, but let us forget this color code, since it is already covered by the RGB profile).

Color profile The used color profile must be included within the mandatory metadata.

Metadata Special metadata must be included and they must not be compressed within the PDF file, so that they are always readable without uncompressing the file.

Dublin Agreement data The metadata concerning the Dublin Agreement have to be included in the proper form.

It is evident that the requirements are pretty stringent even for the less stringent regulation among the various levels of the ISO rules.

With the ScuDo doctoral school dissertation, things are pretty safe, because the preamble example contained in the `toptesi-scudo-example.tex` guarantees a very high success in producing PDF/A-1b compliant files.

The preamble of the document should be as follows[5]:

```
\documentclass[%
   corpo=12pt, % font size
   twoside, % recommended
   tipotesi=scudo,
]{toptesi}
...
\begin{filecontents*}{\jobname.xmpdata}
\Author{Mario Rossi}
\Title{Writing Your Ph.D. Thesis
       with LaTeX}
\Subject{Doctoral dissertations
       in the SCUDO doctoral school}
\Keywords{PDF\sep
          PDF/A\sep
          ISO 19005\sep
          LaTeX\sep
          PhD Thesis\sep
          Engineering\sep
          SCUDO}
\Publisher{Politecnico di Torino}
\end{filecontents*}
...
\usepackage[a-1b]{pdfx}
\ifPDFTeX
  ...
\else
  \usepackage{fontspec}
  \defaultfontfeatures{Ligatures=TeX}
  \setmainfont{Libertinus Serif}
  \setsansfont{Libertinus Sans}
  \setmonofont{Libertinus Mono}%
              [Scale=MatchLowercase]
  \usepackage[math-style=ISO]{unicode-math}
  \setmathfont{Libertinus Math}%
\fi
...
\makeindex[intoc]% configure indexing

\unless\ifcsname ver@hyperref.sty\endcsname
  \usepackage{hyperref}\fi
\hypersetup{%
  pdfpagemode={UseOutlines},
  bookmarksopen,
  pdfstartview={FitH},
  colorlinks,
```

[5] The example file contains a lot of descriptive comments; for brevity here the comments are neglected as well as unnecessary options and commands.

```
46   linkcolor={blue},
47   citecolor={blue},
48   urlcolor={blue}
49 }
50 \includeonly{%
51 ...
52 }
53 %
54 \begin{document}
```

The toptesi-scudo module is conceived in such a way as to preload all the necessary packages; in particular the filecontents package (Pakin, 2018) is preloaded for convenience. In facts the starred or unstarred environment filecontents that is used in lines from 7 to 21 in the above code, normally would not overwrite the file it is supposed to create; therefore, in order to change the environment contents even for the simple correction of a typo, it is necessary to delete the already existing `.xmpdata` file, otherwise the correction does not appear in the output file. With the use of the filecontents package this feature is eliminated and the file is overwritten each time the document is typeset.

The environment output file `\jobname.xmpdata` has this special name that assures that the generated file has the same name as the thesis main file and the specified correct extension.

This file contains the metadata that depend on the specific document; all other necessary metadata are fixed information provided by the pdfx package invoked with the `a-1b` option which specifies the document PDF file should be conformant with the PDF/A-1b regulation.

The `.xmpdata` file contains certain metadata, each one preceded by its keyword; the metadata concerning that example document contain the name of the `Author` as “Mario Rossi” (the Italian equivalent of the Anglo-American John Smith); the main title for the `Title` descriptor and a short but descriptive phrase for the `Subject`; some `Keywords` are also listed separated by the special separator `\sep` required by package pdfx. Of course more specific metadata with other descriptors may be specified; the whole list is found in the documentation of package pdfx.

The lines from 24 to 35 contain the test to discover if the typesetting engine is `pdftex` or something else. We already discussed this feature in subsection 12.1 and here we skip what deals with pdfLaTeX.

As the reader can observe, package pdfx is loaded as the very first one (besides those that are already preloaded); this package loads hyperref (Rahtz and Oberdiek, 2018) with the option pdfa so that certain features of hyperref are modified in order to fulfil the hyperlinking requirements set forth by the ISO rules; this is why in lines 39 and 40 the preamble tests if package hyperref has already been loaded and in case it loads it *without any option*. Such options are set with the command `\hypersetup` that assures to avoid any conflict between the options passed to the package by pdfx and this second possible call. Option clashes produce error messages difficult to correct; so it is better to avoid them in advance.

The text and math fonts loaded with the fontspec facilities are those called Libertinus, a revisited set of OpenType fonts obtained from the Libertine ones; apparently the latter are not maintained any more, and Michael Sharpe reworked, corrected and enriched them and called them with a different, although similar, name; they lack the optical sizes but do a very good job. The math version must be specified after the unicode-math package has been loaded; here I specified the math-style=ISO option because I recommend it very strongly for the ScuDo doctoral theses in engineering disciplines.

19 Conclusion

In this paper I described the problems that arise when typesetting theses in general and how most of these problems have been tackled with the TOPtesi bundle. I entered into the details of the module needed to create typographically pretty nice theses for the ScuDo Doctoral School of Politecnico di Torino. In any case the requirements set forth by the School are all satisfied; all necessary macros and environments have been defined and are available to the authors.

The authors may introduce more macros and environments in order to create their theses. But what is really important, I wish them to be proud of their researches and their results; they will be proud also of the appearance of their typeset results.

References

Ahmetovic, D., T. Armano, M. Berra, C. Bernareggi, A. Capietto, S. Coriasco, N. Murru and A. Ruighi (2018). «Axessibility: creating PDF documents with accessible formulae». *ArsTeXnica*, (26), pp. 50–54.

American Mathematical Society (2013). *The amssymb package*. Version 3.01. Readable with `texdoc amssymb`.

— (2017). *Using the amsthm package*. Version 2.20.3 – Readable with `texdoc amsthm`.

— (2018). *User’s guide for the amsmath package*. Readable with `texdoc amsmath`.

Beccari, Claudio (2018). *The widetable package*. GuIT. Version 1.5. Readable with `texdoc widetable`.

— (2019a). *Il pacchetto TOPtesi*. GuIT. Version 6.2.03. Readable with `texdoc toptesi-it`.

— (2019b). *The extension package curve2e*. GuIt. Version 1.61. Readable with `texdoc curve2e`.

— (2019c). *The TOPtesi bundle*. GuIt. Version 6.3.02. Readable with `texdoc toptesi.pdf` and in Italian `texdoc toptesi-it`.

Carlisle, David (1995). *The indentfirst package*. TUG. Readable with `texdoc indentfirst`.

Chen, Pheong and Michael A. Harrison (2014). «Index preparation and processing». PDF document. The original document is older; this is an updated version by Dan Lueking and Karl Berry. Readable with `texdoc ind`.

C.V. Radhakrishnan, Hàn Thế Thành, Ross Moore and Peter Selinger (2018). *Generation of PDF/X- and PDF/A- compliant PDFs with* `PdfTeX`. River Valley Technologies, Trivandrum, India. Version 1.6 upgraded and valid also for LuaLATEX. Readable with `texdoc pdfx`.

Daly, Patrick W. (2010). *Natural sciences citations and references — Author-year and numerical Schemes*. TUG. Version 8.31b. Readable with `texdoc natbib`.

De Marco, Agostino (2009). «Produrre grafica vettoriale di alta qualità programmando `asymptote`». *ArsTEXnica*, (8), pp. 25–39.

Fear, Simon (2016). *Pubblication quality tables in LATEX*. TUG. Version 1.618033. Readable with `texdoc booktabs`.

Feuersänger, Christian (2018). *Manual for package* pgfplots. TUG. Version 1.16. Readable with `texdoc pgfplots`.

Gässlein, Hubert, Rolf Niepraschk and Josef Tkadlec (2016). *The pict2e package*. TUG. Version 0.3b. Readable with `texdoc pict2e`.

Gregorio, Enrico (2016). «The package imakeidx». PDF document. Version 1.3e, Readable with `texdoc imakeidx`.

Hammerlindl, Andy, John Bowman and Tom Prince (2018). *Asymptote: the vector graphics language*. TUG. Version 244. Readable with `texdoc asymptote`.

Hobby, John D. (2018). *METAPOST– A users' manual*. TUG. Version 2.00 (2.0rc2). Readable with `texdoc metapost`.

ISO 19005-1 (2005). *ISO 19005-1:2005 – Document management – Electronic document file format for long-term preservation*. International Organization for Standardization, Geneva. `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38920`.

ISO 19005-2 (2011). *ISO 19005-2:2011 – Document management – Electronic document file format for long-term preservation*. International Organization for Standardization, Geneva. `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50655`.

ISO 19005-3 (2012). *ISO 19005-3:2012 – Document management – Electronic document file format for long-term preservation*. International Organization for Standardization, Geneva. `http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?ics1=37&ics2=100&ics3=99&csnumber=57229`.

ISO 31/XI (1978). *Mathematical signs and symbols for use in the physical sciences and technology*. International Organization for Standardization, Ginevra. Regulation ISO 31/XI/–1978. Updated with regulation ISO 80000-2: `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=31887`. See also (Thompson and Taylor, 2008).

Lamport, Leslie (1987). *MakeIndex: an Index Processor for LATEX*. TUG. Readable with `texdoc makeindex`.

— (1994). *A document preparation system — LATEX — User's guide and reference manual*. Addison Wesley, Reading, Mass., 2$^{\mathrm{nd}}$ edition.

Lehman, Philip (2018). *The biblatex package – Programmable bibliographies and citations*. TUG. Version 3.12. Readable with `texdoc biblatex`.

Lehman, Philip and Joseph Wright (2018). *The csquotes package — Context sensitive quotation facilities*. TUG. Version 5.2d. Readable with `texdoc csquotes`.

Miklavec, Mojca (2013). *Using* `context` *and tikz terminals for gnuplot in ConTEXt*. Readable with `texdoc gnuplot`. This document gives insructions to install the external program `gnuplot`, version 4.6.0 or later, and to configure it to be used by the typesetting programs of the TEX system.

Milde, Günter (2012). *isomath – Mathematical style for science and technology*. TUG. Readable with `texdoc isomath`.

Oetliker, Tobias (2015). *An acronym environment for LATEX 2ε*. tug. Verison 1.41. Readable with `texdoc acronym`.

Pakin, Scott (2018). *The filecontents package*. TUG. Version 1.4. Readable with `texdoc filecontents`.

RAHTZ, Sebastian and Heiko OBERDIEK (2018). *Hypertext marks in LATEX: a manual for hyperref*. TUG. Readable with `texdoc hyperref`.

ROBERTSON, Will (2019a). *Experimental Unicode mathematical typesetting. Th unicode-math package*. TUG. Version 0.8n. Readable with `texdoc unicode-math`.

— (2019b). *The fontspec package. Font selection for XƎLATEX and LuaLATEX*. TUG. Version 2.7b. Readable with `texdoc fontspec`.

SMITH, Joseph (2018). *siunitx — A comprehensive (SI) units package*. TUG. Version 2.7s. Readable with `texdoc siunitx`.

TALBOT, Nicola L.C. (2019). *User manual for glossaries.sty v4.42*. TUG. Version 4.42. Readable with `texdoc glossaries`.

TANTAU, Till (2019). *TikZ & PGF*. TUG. Version 3.1.1 – Readable with `texdoc tikz` or with `texdoc pgf` or with `texdoc pgfmanual`.

THE LATEX 3 PROJECT TEAM (2018). *The xparse package – Document command parser*. TUG. Readable with `texdoc xparse`.

THOMPSON, Ambler and Barry N. TAYLOR (2008). *Guide for the Use of the International System of Units (SI)*. NIST – National Institute of Standards and Technology. NIST Special Publication 811 – 2008 Edition. Url: `https://www.nist.gov/pml/special-publication-811-extended-contents`.

VEYTSMAN, Boris *et al.* (2019). *nomencl: a package to create a nomenclature*. TUG. Version 5.1. Readable with `texdoc nomencl`.

▷ Claudio Beccari
Emeritus Professor
of Politecnico di Torino
`claudio dot beccari at polito dot it`

Creating accessible pdfs with LaTeX

Ulrike Fischer

Abstract

This article describes the current state and planned actions to improve the accessibility of pdfs created with LaTeX as it is currently undertaken by the LaTeX Team.

Sommario

Questo articolo descrive lo stato attuale e le azioni pianificate per migliorare l'accessibilità dei pdf creati con LaTeX così come garantito dal LaTeXT̃eam.

1 Accessibility of pdf

The pdf language is at its core a *page description* programming language. It describes very accurately how text and graphical elements look and where they are placed on a page. But it doesn't describe the semantical meaning of the elements and the reading order: by looking at the code there is no way to know if a text is a section heading or some watermark or a footnote or if it belongs to a tabular. You can't know where a sentence has its continuation on another page or how many words a text contains, and sometimes it is even impossible to identify the characters: you only see some glyph index number and copy & paste can give gibberish.

This all isn't a problem as long as the pdf is merely meant for printing or viewing but it restricts its use for digital processing like copy & paste, automatic extraction of billing data, reflowing or using the pdf with a screen reader. For such uses you need accessible, structured, extractable content.

«Accessibility» as a standard is described in PDF/UA. It is also included in other standards like PDF/A-1a and PDF/A-2a (the «a» stands for accessible). The standards contain a number of requirements that should help retrieving the content for further processing: for example, that every character has a unicode representation (no gibberish when copy & pasting), that word spaces are correctly marked up (so that reflowing works), that the language of the document and the text is declared (so that a screen reader can guess the pronunciation), that pictures have sensible alternative descriptions. And most importantly: that the document is *tagged*. This last requirement is responsible for adding structure information to the content. It marks up content as section or tabular cell or list item. This improves navigation in the document with, for example, a screen reader, but also exporting to other formats like xml.

TUG maintains a webpage with various links to relevant standards, articles and packages, (see TUG, 2019).

2 Creating a tagged pdf

Tagging consists of two main tasks: at first in the *stream object* of a page every bit of content must be marked and labelled with a number `MCID` n.

The following listing shows a small example. The `BDC` and the `EMC` lines are the start and end markers needed for tagging. The `/H1` indicates that the content is part of a sectioning element.

```
stream
/H1 <<MCID 0>> BDC
BT
/F17 14.3462 Tf 124.802 706.129
 Td [(0.1)-1100(Section)]TJ
ET
EMC
```

In the next step a number of pdf objects must be created to describe the *structure tree*. Every object contains references to the parent `/P` and to one or more kid elements `/K`. The leaf nodes are the `MCID` n created in the first step. A typical object looks roughly like this:

```
5 0 obj
<<
   /Type /StructElem
   /S /H1
   /P 4 0 R
   /K <</Type /MCR /MCID 0>>
>>
endobj
```

This is a structure element of the type `/H1` (and so a sectioning element) and it has one kid element, the text of the section marked above.

Beside this a number of additional settings and objects must be added to the pdf for crossreferencing and «administration».

3 Changing LaTeX

Measured in computer time LaTeX is quite old. LaTeX is not only a format: it was always meant to be extended by packages and classes and over the time many people contributed to LaTeX. It has a quite large user base with *very* varied demands regarding stability, features and development. LaTeX is still used with a variety of engines: pdfTeX, XƎTeX, LuaTeX, (u)pTeX and backends (dvips, dvipdfmx). One could compare LaTeX to an old city: lots of houses built at different times in different styles by various people, some modern, some older, some are in a good state, other are falling apart but nevertheless home to someone.

This means that changing LaTeX is not easy: We can't break lots of packages and old documents even if the reward is accessible pdfs. And we have to consider that documents must be compilable in TeX systems of varying age for example when uploading them to some journal.

A very important aspect of the project long term is to develop a change strategy and manage the integration of core support across the LaTeX universe.

4 First Steps towards tagging

Tagging pdf with LaTeX has been on the agenda for quite some time. Babett Schalitz wrote a thesis about it in 2007, Ross Moore had a number of talks and articles at TUG since then too. When I considered to work on the topic some time ago I got code from both and decided rather quickly that at first some work on the basics was needed. Tagging should in my opinion not be done by creating a package that patches all sorts of commands in other packages: this is much too fragile. It needs proper support in the LaTeX kernel and proper support in the main classes and packages. I also thought that to identify the needed support and to test implementations and interfaces concrete code was needed. So I wrote the package tagpdf. The package offers core commands to tag a document and to activate some of the other requirements needed to make a pdf accessible. The low-level code to mark up a text as a section looks roughly like this:

```
\tagstructbegin{tag=H1}
  \tagmcbegin{tag=H1}
    Section
  \tagmcend
\tagstructend
```

The `\tagstructXX` commands create the structure, while the `\tagmcXX` commands add the `MCID` marks to the page stream.

The tagpdf package currently works with PDFLaTeX and LuaLaTeX – with lualatex the results are the best as one doesn't have to worry about the behaviour at page breaks – but with the help of the work on the pdfresources project described below it should be possible to extend it to other engines and backends.

5 LaTeX-dev

Another important step towards accessible pdfs was the implementation of the latex-dev format by the LaTeX team and the maintainers of TeX-Live and MiKTeX: latex-dev is a pre-release of LaTeX from the development branch and made available on CTAN. It allows users of a current TeX distribution to test their documents and code against the upcoming LaTeX release by simply using a latex-binary with the addition `-dev` attached.

latex-dev has not been created solely with tagging in mind but it will help us to coordinate and test changes with package and class authors and so it is an important part of the project.

6 Pdf resource management

When tagging a pdf one has to add a number of settings to pdf dictionaries which can be described as «global resources». As already mentioned in an answer (Oberdiek, 2015), LaTeX has no interfaces for this:

> Unhappily, the LaTeX format has overslept the PDF development quite entirely. Managing global resources is the prime task for an OS, format in TeX speek. Because of the missing resource manager, both [tikz and transparent] packages do what most packages do, they think they are alone and add their stuff to the resource, . . .

With tagging entering the scene it was clear that something needed to be done to remedy this problem and so the pdfresource project in the LaTeX github was created: it contains a (still quite experimental) expl3-style which offers commands to add contents to pdf resources in a controlled way. It also offers backend independent interfaces to a number of core commands needed when writing objects to a pdf. The package works with the main engines (PDFTeX, LuaTeX and XeTeX) and backends (dvipdfmx and – more or less – dvips).

The main task for the next months is to test the code, to integrate it into the kernel and to adapt existing packages to use it. The number of packages which should use the pdf resource manager is not very large but it includes important packages like hyperref, tikz, media9, pdfx.

7 Adapting the engines

Another open issue that emerged during the last year was missing functionalities in engines and backends. For example PDFTeX is not ready for pdf 2.0: it has no command to set a major pdf version. As pdf 2.0 adds important features needed for accessibility (the concept of associated files) this is clearly something that should be changed. It would be also useful if PDFTeX could execute code at shipout time as it can be done with luatex with `\latelua`. The dvipdfmx backend and dvips are missing additional color stacks.

8 Adding hooks

As already shown in section 2 and 4, tagging a pdf requires adding quite a number of commands. Obviously all the standard structures should if possible add the needed code automatically. For this hooks are needed at the right places. The «right place» has firstly a technical meaning: with the exception of LuaTEX, the tagging code inserts *whatsits*; this means it can change the output if used in the wrong place (as sometimes anchors set by hyperref do).

But more importantly the «right place» means that we need to identify the *owner* of the code which should insert the tagging code. For example sections are generally created with `\@startsection`. So this kernel command looks like a natural place to insert hooks for tagging commands. On the other hand chapters and parts have special commands created by the classes. Does it make sense if the kernel handles the one part and the classes the other? Another example are bibliographies and glossaries: packages like biblatex and glossaries look like the natural owner here – and both packages have already lots of hooks which make it easy to implement tagging – but both also use standard structures like lists or tabulars and additions to this generic environments could clash with their needs.

This means that beside a pdf resource manager we also need a hook management. And we need lots of real use cases and examples to be able to investigate the various dependencies.

9 Mathematics

How to tag maths is still an open problem. There are quite a number of possibilities to make it accessible.

- One is to attach the LATEX source code either as file or verbatim with `/Actualtext` to the math structure. For a number of environments this can be automated quite well as the axessibility package demonstrate (but it is difficult for inline math input with `$..$`). The usability with a screen reader is not bad – even if not every word was correctly read aloud in my tests – but it requires that the user understands LATEX input syntax and with large equations and complicated grouping it can be quite difficult to follow and to navigate through subequations. The usability can be improved if one invests the time to manually split the math and add explaining words.

- Another possibility is to mark all the maths bits with mathml structure names. At least with LuaTEX this can probably be done more or less automatically – proof of concepts are the ConTEXt format and TEX4ht. But it is unknown wether screen readers or other applications can actually use the information.

- A third possibility is to convert the equation to mathml, for example with mathjax, and attach it as associated file to the structure. But here too it is unclear how such a mathml can be processed by the pdf consumer. It is also unknown which flavour of mathml should be used in this case.

The pdf standard requires that glyphs and symbols are mapped to unicode. Here too variants are possible. a could be mapped to U+1D44E (Mathematical Italic Small A) or U+0061 (Latin Small Letter A), $\int$ could mapped to U+222B (Integral) or to `\int` (as it is done by the package mmap). The first alternative sounds more unicode-like but actually the screen readers don't seem to know what to do with the symbols.

The main task here is to get more information to be able to decide about which route to follow.

10 Contacts

Quite a number of questions and projects circle around the pdf specification, the needs of users and of pdf consumer applications. To get tagging working it is not enough to know how TEX works. So one important part of the tagging project is to get in contact with people having inside knowledge about pdf and pdf consumer applications in various pdf related organizations and to promote the project in the TEX world to get user feedback.

11 Summary

Adding tagging facilities to LATEX is a large project with many aspects. Happily it doesn't have to be done in one large jump: with the tagpdf package is it already possible for adventurous users with a bit of knowledge in TEX programming to tag quite large documents. Despite the clear warning in the documentation that it isn't meant for production I already got a number of feedbacks of successful uses. This gives hope that it can evolve to a stable and usable system.

References

Oberdiek, Heiko (2015). «tikz and transparent incompatibility». https://tex.stackexchange.com/a/253417/2388.

TUG (2019). «Pdf accessibility and pdf standards». https://www.tug.org/twg/accessibility/.

▷ Ulrike Fischer
LATEX Project
Mönchengladbach
fischer at troubleshooting-tex dot de

Axessibility 2.0: creating tagged PDF documents with accessible formulae

D. Ahmetovic, T. Armano, C. Bernareggi, A. Capietto, S. Coriasco, B. Doubrov, A. Kozlovskiy, N. Murru

Abstract

PDF documents containing formulae generated by LaTeX are usually not accessible by assistive technologies for visually impaired people (i.e., by screen readers and Braille displays). The LaTeX package `axessibility.sty` that we developed manages this issue, allowing to create PDF documents where the formulae are read by such assistive technologies, through the insertion of hidden comments. In this paper we describe the evolution of the package, that in the latest version automatically generates also the tagging of the formulae. The package however does not generate documents tagged according to the PDF/UA standard.

Sommario

I documenti PDF contenenti formule generati da LaTeX non sono solitamente accessibili mediante tecnologie assistive per persone con disabilità visive (i.e., screen reader e display Braille). Il pacchetto LaTeX `axessibility.sty` da noi sviluppato risolve questo problema, permettendo di creare documenti PDF in cui le formule vengono lette da tali tecnologie assistive, tramite l'inserimento di commenti nascosti. In questo articolo descriviamo l'evoluzione del pacchetto, che nella più recente versione genera automaticamente anche il tagging delle formule. Il pacchetto però non genera documenti etichettati secondo lo standard PDF/UA.

1 Introduction

PDF documents are widely used to digitally publish scientific content, such as papers or textbooks. Mathematical formulae, frequently contained within such documents, are not accessible by screen reader users because they are commonly rendered as bi-dimensional images. The burden of making digital documents accessible to visually-impaired persons is often left to the document author, who needs to provide descriptions for each visual content in the form of alternate text. This procedure is time consuming, error-prone and it needs to be done by a sighted person. Additionally, in the case of mathematical formulae, a verbal description does not provide the same information as the original mathematical notation. In many cases no alternate text is even provided because authors are not aware of the accessibility needs of screen reader users.

In this paper, we show the features of the package `axessibility.sty` (whose first version is also described in Armano *et al.* (2018)) that provides the first method for an automatised production of accessible PDF documents with mathematical contents through LaTeX. We would like to highlight that this package does not produce fully tagged PDF, such as the standard PDF/UA, but it allows to obtain a PDF where formulae are marked and described using the `/Alt` and `/ActualText` attributes.

2 Related Work

Assistive technologies for people with visual impairments (e.g., screen readers, Braille displays, magnifiers) are used effectively and proficiently to read and edit digital documents containing structured text. Instead, still many accessibility issues remain for what concerns documents including mathematical formulae and images (e.g., diagrams, graphs, technical drawings; Archambault *et al.* (2007); Armano *et al.* (2014)). A number of studies have been conducted to improve non-visual access to scientific content, mainly along two research lines: to facilitate editing of scientific documents through non-visual tools, and to enable people with sight impairments to read scientific documents in digital formats.

The former research work has led to different multimodal systems that are now available to author scientific documents through non-visual tools. For instance, the LAMBDA editor (Bernareggi, 2010) is used mostly by blind people to write and process text and mathematical formulae through Braille display and speech output. This system adopts a sequential code to represent mathematical notation, specifically designed for blind people and usable only in this editor. Hence, it has got widespread only among some communities of blind people and it cannot become a mainstream tool to produce accessible scientific content by sighted people, too. A different approach consists in editing LaTeX documents through speech and Braille support (Pepino *et al.*, 2006; Melfi G., 2018; Yamaguchi *et al.*, 2008; Manzoor *et al.*, 2018, 2019; Sorge, 2016). This approach has the advantage to rely on LaTeX, which is a de facto standard for authoring scientific documents. Unfortunately, since these tools are produced for a small community, due to the rapid evolution of technology, they often incur in maintainance and compliance issues.

For what concerns reading digital scientific documents, many studies have been undertaken to create non-visual reading tools for the most widespread digital formats. In particular, research has focused on web publishing Microsoft Word, LATEX and PDF documents. In recent years, mathematical content has been published on the web through images of formulae, by embedding MathML in the web page or through MathJax, a JavaScript display engine for mathematical formulae. Images of formulae are inaccessible to screen readers, hence they can be adapted to be read by screen readers only through a proper alternative text (e.g., the LATEX equivalent). On the contrary, MathML and MathJax can be used to create accessible web pages. MathML, especially the content markup, can be interpreted by most common screen readers to generate a verbal description of the formula (Bernareggi and Archambault, 2007; Sorge *et al.*, 2014). Moreover, MathPlayer, a web browser plug-in for rendering MathML on the screen, through speech output and on Braille devices, enables hierarchical navigation of mathematical formulae, including bi-dimensional notations such as matrices (Soiffer, 2018). MathJax can be embedded in web pages making available adaptable accessibility features for representing and navigating formulae (e.g., LATEX, ASCIIMath or CSS representation; Cervone *et al.* (2016); Cervone and Sorge (2019)). Taking Microsoft Word into account, mathematical formulae can be read by the speech synthesizer or on a Braille display through MathPlayer. Nonetheless, due to the visual features of Microsoft Word, interaction with screen readers is often not easy. LATEX documents can be read by people with sight impairments either reading the source file on the Braille display or through editors that support speech reading of LATEX (e.g., ChattyInfty by Science Access Net; Pepino *et al.* (2006); Melfi G. (2018); Yamaguchi *et al.* (2008); Manzoor *et al.* (2019)). Furthermore, also converters from LATEX to some national Braille codes for mathematics are available (Papasalouros and Tsolomitis, 2017). Since national Braille codes can represent only a limited amount of mathematical notations, these converters can transform only a subset of the source LATEX document.

For PDF files, frequently used as a medium for publishing digital scientific documents, the accessibility of mathematical content has been developed in the scope of the so-called Tagged PDF, which embeds the document semantics directly into the visual representation of the page. Both ISO 32000-1:2008 (specifying PDF 1.7) and the recent ISO 32000-2:2017 (for PDF 2.0) suggest the use of MathML syntax for describing the semantics of mathematical formulae. In addition, PDF 2.0 standard opens the door for any alternative syntax (for example, the original LATEX representation of the formula), which can be associated with any structure element in Tagged PDF. However, due to the novelty of this approach, it is not yet supported by the screen readers and, thus, may be considered only in the long-term scope.

Another approach widely supported by the majority of the screen readers is to add accessibility features to mathematical content as alternate text. It can be specified manually using, for example, a proprietary editor such as Adobe Acrobat. Guidelines have been produced to create accessible PDF according to this procedure (Uebelbacher *et al.*, 2014) with a focus on mathematical content (Moore, 2009, 2014; Borsero *et al.*, 2016).

However, this approach requires the availability of a suitable editor, and it entails additional labor from the document author. Furthermore, alternate text most often does not carry the same semantic value as the original mathematical content. Yet another approach consists in transforming PDF files into LATEX or HTML+MathML documents by performing OCR (Baker *et al.*, 2010; Suzuki and Yamaguchi, 2017). However, the resulting document has to be proofread because of possible recognition errors. Proofreading process is usually time consuming and it has to be done by a sighted person who can compare the PDF document with the OCR result.

3 The axessibility LATEX package

We provided a solution to the problem described above through our package axessibility, see, e.g., Ahmetovic *et al.* (2018a,b); Armano *et al.* (2018). In its most recent version, release 2.0, which will soon be available in CTAN, we employed the `tagpdf` package, created by Ulrike Fischer (see Fischer (2019)), replacing the `accsupp` package, on which the 1.x versions of the axessibility package relied. The package implements insertion of the original LATEX formulae as properties of the Span elements containing visual representation of the mathematical content in the resulting PDF document, by means of the commands provided by the `tagpdf` package.

In more detail, each inline or display formula in the source LATEX document is wrapped into a marked content sequence (see the documentation of the `tagpdf` package for more details on the difference between structure elements and marked content sequences in Tagged PDF). In addition, the original formula is added to this marked content sequence as `/ActualText` and `/AltText`. These properties are read by screen readers and braille displays instead of the `ASCII` representation of the formula, which is often incorrect. Additionally, the package adds a minimal Tagged PDF structure to the output PDF. This includes at the moment the top level Document structure element to mark the beginning and the end of the document and the P (paragraph) tag for each formula. Further extension of this set of tags (like automatic tagging of all paragraphs, section headers, etc) is still a work in progress. For details about the structure of a PDF document, we refer to the ISO standards 32000-1:2008 (2008); 32000-2:2017 (2017).

As the tagpdf package, the axessibility 2.0 package is currently experimental and it is aimed for individual tests and experiments.

3.1 Usage

To create an accessible PDF document for visually impaired people, the authors just need to include the axessibility package into the preamble of their LaTeX project. The supported mathematical environments will then automatically produce the /ActualText and /AltText contents and include them in the produced PDF file. Formulae will also be automatically tagged, as well as the document environment. The tagging of other text tokens (paragraphs, sections, etc.), at the moment, has to be inserted manually, under the guidelines of the tagpdf package.

The environments for writing formulae which are presently supported are \(, \[, equation*, equation, align*, and align. Hence, any formula inserted using one of these environments is accessible and tagged in the corresponding PDF document. The click-copy of the formula LaTeX code from the PDF reader, to be pasted elsewhere, is presently not working with this new release.

Inline and displayed mathematical modes activated by the old syntaxes \$...\$ and \$\$...\$\$ are not supported by the axessibility package (as in the previous versions). However, external scripts provided as companion software can address, at some extent, the problem of source files where the old TeX syntax is used (see Section 4 below).

Below, an example of LaTeX code, illustrating the usage of axessibility, jointly with tagpdf.

```
\documentclass{article}
\usepackage{etoolbox,axessibility}

\begin{document}

\tagstructbegin{tag=P}
  \tagmcbegin{tag=P}
        A simple displayed formula:
  \tagmcend
\tagstructend

\begin{equation*}
x=\frac{3a^2}{n+m}
\end{equation*}

\tagstructbegin{tag=P}
  \tagmcbegin{tag=P}
        A multiline formula, aligned,
           with label:
  \tagmcend
\tagstructend
\begin{align}
70xy^2+105x^2y-35xy7
& = 35\left(2xy^2+3x^2y-xy7\right) =
    \\
& = 35x\left(2y^2+3xy-y7\right) =
    \\
& = 35xy\left(2y+3x-7\right)
\end{align}

\end{document}
```

We observe that, in these cases, the author can write the formulae without adding anything else. Moreover, inside the source code of the PDF file, we find /ActualText and /AltText contents, with the (Hex) LaTeX code inside, automatically generated by the axessibility.sty package, as well as the equation tags, namely:

```
/P
<</MCID 1
/Alt <FEFF00200020078003D005C
      0066007200610063002007B
      00330061005E0032007D007B
      006E002B006D007D0020>
/ActualText <FEFF00200020078003D005C
             0066007200610063002007B
             00330061005E0032007D007B
             006E002B006D007D0020>


>>
```

and

```
/P
<</MCID 3
/Alt <FEFF00370030000780079005E
       0032002B00310030000350078
       005E00320079002D00330035
       00780079003700200260020
       003D00200033003500 5C006C
       00650066007400200280032
       00780079005E0032002B0033
       0078005E00320079002D0078
       00790037005C007200690067
       0068007400200029002 0003D
       0020005C005C00200260020
       003D00200033003500 78005C
       006C006500660074002 00028
       00320079005E0032002B0033
       00780079002D00790037005C
       00720069006700680074002 0
       00290020003D0020005C005C
       00200026002 0003D00200033
       00350078007 9005C006C0065
       00660074002 0002800320079
       002B00330078002D0037005C
       00720069006700680074002 0
       0029>
/ActualText <FEFF00370030000780079005E
       0032002B00310030000350078
       005E00320079002D00330035
```

```
              0078007900370020002600 20
              003D002000330035005C006C
              006500660074002000280032
              00780079005E0032002B0033
              0078005E00320079002D0078
              00790037005C007200690067
              006800740020002900200003D
              0020005C005C002000260020
              003D00200033003500 78005C
              006C00650066007400200028
              00320079005E0032002B0033
              00780079002D00790037005C
              007200690067006800740020
              00290020003D0020005C005C
              00200026002000 3D00200033
              00350078007900 5C006C0065
              006600740020002800320079
              002B00330078002D0037005C
              007200690067006800740020
              0029>
 >>
```

respectively. Here the /Alt and /ActualtText keys are followed by the UTF-16 encoded values in the Hexadecimal format. So, this makes our solution fully Unicode compliant.

We note that such use of /Alt and /ActualText keys is not fully aligned with the best practices of PDF accessibility techniques. But it does open the door for real world tests and further experiments. In particular, the screen reader will read correctly the LATEX commands. Moreover, the JAWS and NVDA dictionaries that we created provide the reading in the natural language, in the case that the user does not know the LATEX commands. It is strongly recommended to use the most recent version of tagpdf (available through the *GitHub* website), as well as the most updated versions of the TexLive distribution.

3.2 Technical Overview

In axessibility we first load the requested packages, configure tagpdf, and define a pair of internal variables.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{axessibility}

\RequirePackage{tagpdf}
\tagpdfsetup{tabsorder=structure,
    uncompress,activate-all,
    interwordspace=true}
\tagpdfifpdftexT
 {
\pdfcompresslevel=0
  %set language / can also be done
      with hyperref
  \pdfcatalog{/Lang (en-US)}
  \usepackage[T1]{fontenc}
  \input glyphtounicode
  \pdfgentounicode=1
 }
\tagpdfifluatexT
 {
  %set language / can also be done
      with hyperref
  \pdfextension catalog{/Lang (en-US)}
  \RequirePackage{fontspec}
  \RequirePackage{luacode}
  \newfontface\zerowidthfont{freeserif
      }
\directlua{
pdf.setcompresslevel(0)
pdf.setmajorversion(2)
pdf.setminorversion(0)
}
}

\RequirePackage{amsmath}
\RequirePackage{amssymb}
\RequirePackage{xstring}

\newtoks\@mltext
\newtoks\@mltexttmp
```

Then, we redefine the document environment, so that the PDF file is automatically tagged at the Document level.

```
\makeatletter
\let\begin@document=\document
\let\end@document=\enddocument
\renewcommand{\document}{\
    begin@document\tagstructbegin{tag=
    Document}}
\renewcommand{\enddocument}{\
    tagstructend\end@document}
\makeatother
```

Subsequently, we redefine the inline formula environment, to make it accessible, inserting its (hidden) LATEX code. We also define an internal command to produce a space (which is useful in passing parameters to some of our redefined environments).

```
\makeatletter
\newenvironment{temp@env}{%
  \relax\ifmmode\@badmath\else$\fi%
 \collect@body\wrap}{%
  \relax\ifmmode\ifinner$\else\
      @badmath\fi\else \@badmath\fi}
\protected\def\(#1\){\begin{temp@env
    }#1\end{temp@env}}
\makeatother

\newcommand{\auxiliaryspace}{ }
```

The core of the package is represented by the wrapping procedures. The first one, \wrap, is used for both the inline, as well as the displayed single line, formulae environments (numbered and unnumbered), which we redefine in order to obtain their automatic tagging and insertion of the corresponding LATEX code in the /ActualText and /AltText contents. The wrapper receives as parameter the code within the environment, obtained by means of the \collect@body command (from the amsmath package), and passes it to the tagging commands defined in tagpdf.

```
\makeatletter
\long\def\wrap#1{
\tagstructbegin{tag=P,alttext-o=\
    detokenize\expandafter{#1},
    actualtext-o=\detokenize\
    expandafter{#1}}
 \tagmcbegin{tag=P,alttext-o=\
     detokenize\expandafter{#1},
     actualtext-o=\detokenize\
     expandafter{#1}}
 #1
 \tagmcend
\tagstructend
}
\makeatother

\makeatletter
\renewenvironment{equation}{%
 \incr@eqnum
  \mathdisplay@push
  \st@rredfalse \global\@eqnswtrue
 \mathdisplay{equation}%
  \collect@body\wrap\auxiliaryspace}{%
  \endmathdisplay{equation}%
  \mathdisplay@pop
  \ignorespacesafterend
}
\makeatother

\makeatletter
\renewenvironment{equation*}{%
  \mathdisplay@push
  \st@rredtrue \global\@eqnswfalse
  \mathdisplay{equation*}%
  \collect@body\wrap\auxiliaryspace}{%
  \endmathdisplay{equation*}%
  \mathdisplay@pop
  \ignorespacesafterend
}
\makeatother

\makeatletter
\protected\def\[#1\]{\begin{equation
    *}#1\end{equation*}}
\makeatother
```

The next two procedures, \wrapml and \wrapmlstar, perform the same task for the multiline environments. We need a different routine here, due to the more involved typesetting procedure of multiline environments like align and align*, which are likewise redefined.

```
\makeatletter
\long\def\wrapml#1{
\def\@mltext{\detokenize\expandafter
    {#1}}
\def\@mltexttmp{}
\StrBehind[6]{\@mltext}{ }[\@mltexttmp
    ]
\StrGobbleRight{\@mltexttmp}{1}[\
    @mltext]
\tagstructbegin{tag=P,alttext-o=\
    detokenize\expandafter{\@mltext},
    actualtext-o=\detokenize\
    expandafter{\@mltext}}
 \tagmcbegin{tag=P,alttext-o=\
     detokenize\expandafter{\@mltext},
     actualtext-o=\detokenize\
     expandafter{\@mltext}}
 #1
}
\makeatother

\makeatletter
\long\def\wrapmlstar#1{
\def\@mltext{\detokenize\expandafter
    {#1}}
\def\@mltexttmp{}
\StrBehind[5]{\@mltext}{ }[\@mltexttmp
    ]
\StrGobbleRight{\@mltexttmp}{1}[\
    @mltext]
\tagstructbegin{tag=P,alttext-o=\
    detokenize\expandafter{\@mltext},
    actualtext-o=\detokenize\
    expandafter{\@mltext}}
 \tagmcbegin{tag=P,alttext-o=\
     detokenize\expandafter{\@mltext},
     actualtext-o=\detokenize\
     expandafter{\@mltext}}
 #1
}
\makeatother

\makeatletter
\renewenvironment{align}{%
  \collect@body\wrapml\auxiliaryspace
 \start@align\@ne\st@rredfalse\m@ne
}{%
  \math@cr \black@\totwidth@
  \egroup
 \ifingather@
    \restorealignstate@
    \egroup
    \nonumber
    \ifnum0=`{\fi\iffalse}\fi
  \else
    $$%
  \fi
  \ignorespacesafterend
  \tagmcend
  \tagstructend
}

\renewenvironment{align*}{%
  \collect@body\wrapmlstar\
      auxiliaryspace
  \start@align\@ne\st@rredtrue\m@ne
}{%
  \endalign
}

\makeatother

\endinput
```

We are presently working to make \wrapml and \wrapmlstar more flexible, so that they will work correctly with all the other multiline environments provided by the amsmath package. This will make all of them accessible and tagged, as those illustrated above. At the moment, the package works correctly when typesetting with both PDFLaTeXas well as LuaLaTeX.

4 Supporting Software

In addition to the axessibility package, we developed additional software to address two use cases: 1) Preprocessing Scripts for the application of axessibility on existing documents, and 2) Screen Reader Dictionaries for natural language reading of formulae made accessible with axessibility. We are currently working on these supporting software, to fix some of the issues we detected through user's reports and suggestions, and to expand their applicability range.

4.1 Preprocessing Scripts

axessibility restricts the syntax that can be used to write mathematical formulae to specific environments and math mode syntax. Instead, existing documents may contain unsupported syntax, and therefore cannot be used with axessibility without being first opportunely edited. We provide *Axesscleaner*, an external script written in Python and Perl, through which it is possible to substitute unsupported commands and environments with suitable replacements, thus enabling the use of axessibility on existing LaTeX documents.

An additional issue lies in the usage of user-defined macros in the LaTeX code. While this is a common practice to avoid code repetitions and simplify document authoring, it can limit the accessibility of formulae with axessibility. Indeed, axessibility is transparent to commands used in math environments, which means that it will include standard LaTeX as well as custom macros within the PDF replacement text. However, custom commands used by an author may bear no meaning for other readers. Thus, *Axesscleaner* also replaces user defined macros with their content, in order to only contain standard LaTeX code within the PDF replacement text.

4.2 Screen reader dictionaries

Mathematical formulae included as PDF replacement text using axessibility are easy to read by LaTeX proficient users, using either a screen reader or a braille display. However, for novice users, the LaTeX code read by a screen reader may be difficult to comprehend.

To address this problem, we also provide dictionaries for *NVDA* and *JAWS* screen readers, which convert LaTeX commands contained within the PDF replacement text created by axessibility into their natural language counterparts (*e.g.*, '\frac{2}{3}' becomes "two thirds"). We are currently developing additional screen reader scripts to enable interactive navigation of formulae, and we are exploring more sophisticated natural language processing techniques to personalize formula reading considering their complexity and context, as well as user's proficiency with math.

5 Acknowledgements

The authors wish to thank the several volunteers with visual impairment who provided their fundamental contribution.

References

32000-1:2008, ISO (2008). «Document management - Portable document format - Part 1: PDF 1.7». International standard, ISO. Https://www.iso.org/standard/51502.html.

32000-2:2017, ISO (2017). «Document management - Portable document format - Part 2: PDF 2.0». International standard, ISO. Https://www.iso.org/standard/63534.html.

Ahmetovic, Dragan, Tiziana Armano, Cristian Bernareggi, Michele Berra, Anna Capietto, Sandro Coriasco, Nadir Murru, Alice Ruighi and Eugenia Taranto (2018a). «Axessibility: a LaTeX Package for Mathematical Formulae Accessibility in PDF Documents». In *Conference on Computers and Accessibility*. ACM.

Ahmetovic, Dragan, Tiziana Armano, Michele Berra, Cristian Bernareggi, Anna Capietto, Sandro Coriasco, Nadir Murru and Alice Ruighi (2018b). «Axessibility: creating PDF documents with accessible formulae». *ArsTeXnica*, (26), pp. 50–54. https://www.guitex.org/home/it/numero-26-ottobre-2018.

Archambault, D., B. Stoger, D. Fitzpatrick and K.: Miesenberger (2007). «Access to scientific content by visually impaired people». *Upgrade*.

Armano, T., A. Capietto, M. Illengo, N. Murru and R. Rossini (2014). «An overview on ict for the accessibility of scientific texts by visually impaired students». In *SIREM/SIE-L Conference*.

Armano, T., A. Capietto, S. Coriasco, N. Murru, A. Ruighi and E. Taranto (2018). «An automatized method based on LaTeX for the realization of accessible PDF documents containing formulae». In *Proc. ICCHP*. Lecture Notes in Computer Science, Springer.

Baker, Josef B., Alan P. Sexton and Volker Sorge (2010). «Faithful Mathematical Formula Recognition from PDF Documents». In

Proceedings of the 9th IAPR International Workshop on Document Analysis Systems. ACM, New York, NY, USA, DAS '10, pp. 485–492. http://doi.acm.org/10.1145/1815330.1815393.

Bernareggi, C. (2010). «Non-sequential mathematical notations in the LAMBDA system». In *Proc. ICCHP*. Springer.

Bernareggi, C. and D. Archambault (2007). «Mathematics on the web: emerging opportunities for visually impaired people». In *Conference on Web accessibility*. ACM.

Borsero, M., N. Murru and A. Ruighi (2016). «Il LaTeX come soluzione al problema dell'accesso a testi con formule da parte di disabili visivi». *ArsTeXnica*. https://www.guitex.org/home/it/numero-22-ottobre-2016.

Cervone, Davide and Volker Sorge (2019). «Adaptable Accessibility Features for Mathematics on the Web». In *Proceedings of the 16th Web For All 2019 Personalization - Personalizing the Web*. ACM, New York, NY, USA, W4A '19, pp. 17:1–17:4. http://doi.acm.org/10.1145/3315002.3317567.

Cervone, Davide, Peter Krautzberger and Volker Sorge (2016). «Towards Universal Rendering in MathJax». In *Proceedings of the 13th Web for All Conference*. ACM, New York, NY, USA, W4A '16, pp. 4:1–4:4. http://doi.acm.org/10.1145/2899475.2899494.

Fischer, U. (2019). «The tagpdf package, v0.61». *CTAN repository*. https://ctan.org/pkg/tagpdf.

Manzoor, Ahtsham, Murayyiam Parvez, Suleman Shahid and Asim Karim (2018). «Assistive Debugging to Support Accessible LaTeX Based Document Authoring». In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, New York, NY, USA, ASSETS '18, pp. 432–434. http://doi.acm.org/10.1145/3234695.3241013.

Manzoor, Ahtsham, Safa Arooj, Shaban Zulfiqar, Murayyiam Parvez, Suleman Shahid and Asim Karim (2019). «ALAP: Accessible LaTeX Based Mathematical Document Authoring and Presentation». In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, CHI '19, pp. 504:1–504:12. http://doi.acm.org/10.1145/3290605.3300734.

Melfi G., Stiefelhagen R., Schwarz T. (2018). «An Inclusive and Accessible LaTeX Editor». In *Proc. ICCHP*. Lecture Notes in Computer Science, Springer.

Moore, R.: (2009). «Ongoing efforts to generate tagged PDF using pdfTEX». *TUGboat, Vol.30, No 2*.

— (2014). «PDF/A-3u as an Archival Format for Accessible Mathematics». In *Watt*, CICM.

Papasalouros, A. and A.: A Tsolomitis (2017). «Direct TeX-to-Braille transcribing method». *Science Education for Students with Disabilities*.

Pepino, Alessandro, Corinna Freda, Fiorentino Ferraro, S Pagliara and Francesco Zanfardino (2006). «"BlindMath" a new scientific editor for blind students». In *Proc. ICCHP*. Lecture Notes in Computer Science, Springer.

Soiffer, N. (2018). «Mathplayer: web-based math accessibility». In *Conference on Computers and Accessibility*. ACM.

Sorge, Volker (2016). «Supporting Visual Impaired Learners in Editing Mathematics». In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, New York, NY, USA, ASSETS '16, pp. 323–324. http://doi.acm.org/10.1145/2982142.2982212.

Sorge, Volker, Charles Chen, T. V. Raman and David Tseng (2014). «Towards Making Mathematics a First Class Citizen in General Screen Readers». In *Proceedings of the 11th Web for All Conference*. ACM, New York, NY, USA, W4A '14, pp. 40:1–40:10. http://doi.acm.org/10.1145/2596695.2596700.

Suzuki, Masakazu and Katsuhito Yamaguchi (2017). «ChattyBooks and ChattyBook Service». In *Proceedings of the 14th Web for All Conference on The Future of Accessible Work*. ACM, New York, NY, USA, W4A '17, pp. 30:1–30:2. http://doi.acm.org/10.1145/3058555.3060619.

Uebelbacher, A., R. Bianchetti and M. Riesch (2014). «Pdf Accessibility Checker (PAC 2): The First Tool to Test PDF Documents for PDF/UA Compliance». In *Proc. ICCHP*. Lecture Notes in Computer Science, Springer.

Yamaguchi, Katsuhito, Toshihiko Komada, Fukashi Kawane and Masakazu Suzuki (2008). «New features in math accessibility with infty software». In *International Conference on Computers for Handicapped Persons*. Springer, pp. 892–899.

▷ D. Ahmetovic
Dipartimento di Informatica,
Università degli Studi di Milano
`dragan dot ahmetovic at unito dot it`

▷ T. Armano
Dipartimento di Matematica "G. Peano",
Università degli Studi di Torino
`tiziana dot armano at unito dot it`

▷ C. Bernareggi
Dipartimento di Informatica,
Università di Milano
`cristian dot bernareggi at unimi dot it`

▷ A. Capietto
Dipartimento di Matematica "G. Peano",
Università degli Studi di Torino
`anna dot capietto at unito dot it`

▷ S. Coriasco
Dipartimento di Matematica "G. Peano",
Università degli Studi di Torino
`sandro dot coriasco at unito dot it`

▷ B. Doubrov
Dual Lab, Belgium
`boris dot doubrov at duallab dot com`

▷ A. Kozlovskiy
Dual Lab Bel, Belarus
`k dot sasha1994 at gmail dot com`

▷ N. Murru
Dipartimento di Matematica "G. Peano",
Università degli Studi di Torino
`nadir dot murru at unito dot it`

Uno script bash di ausilio alla redazione di manoscritti

Gianluca Pignalberi

Sommario

La fase di redazione di manoscritti ci pone spesso di fronte a una serie di cattive pratiche reiterate dagli autori. La correzione interamente manuale può essere fonte di dimenticanze. Vediamo come uno script bash ci consente di minimizzarle.

Abstract

A manuscript editing session puts us in front of a series of authors' repeated bad practices. An entirely-by hand correction can be source of oversights. We will see how a bash script allows us to minimize them.

1 Introduzione

Il mio lavoro di impaginatore LaTeX mi mette molto più spesso di fronte a manoscritti redatti con un word processor che a manoscritti redatti con LyX o direttamente in LaTeX.[1] In base a quanto visto direttamente, è raro che gli utenti di word processor mantengano alta la propria attenzione a qualcosa che non sia il contenuto quando redigono i proprî manoscritti. La loro negligenza relativa alle regole tipografiche più elementari e logiche fa sì che ai redattori si presentino sovente dei documenti dall'aspetto rabberciato e povero, quando non confuso e incoerente, dal punto di vista tipografico.

1. Molti autori usano scorrettamente i glifi e ciò danneggia il testo agli occhi dei redattori e dei lettori più esigenti e pignoli.

2. Altri, non necessariamente diversi, non fanno caso all'interezza delle porzioni di testo alle quali applicano determinate proprietà. Ciò può introdurre dei problemi nell'eventuale ebook, oltre ad aumentare la dimensione del file `.tex` risultante dalla conversione.

3. Molti autori tendono a non usare gli stili: questo può far sì che l'aspetto di elementi di pari semantica non sia coerente.

4. Molti autori, o forse il word processor da essi utilizzato, tendono a impostare incoerentemente più lingue all'interno dei proprî documenti.

Il lavoro del redattore è già messo a dura prova dalla correzione del testo: come esposto in Rawlinson (1976) ed esemplificato in Polidoro (2012), il cervello umano è in grado di leggere correttamente le parole con la prima e l'ultima lettera poste correttamente e le altre mescolate e magari con qualche refuso, fungendo così da correttore ortografico e rendendo difficile la correzione delle bozze.[2] Quando si arriva al momento di controllare tutte quelle piccole minuzie relative alle più elementari regole tipografiche, tale lavoro può diventare ancora peggiore. Due famosi studî psicologici (Simons e Levin (1998) e Simons e Chabris (1999)) hanno mostrato come l'attenzione umana nei confronti di un compito ci renda ciechi ai cambiamenti anche macroscopici del mondo circostante. Posso supporre, non essendo stato in grado di reperire studî specifici, che anche l'attenzione selettiva, oltre alla capacità di "correzione inconscia" che forse funziona anche sui simboli, influisca negativamente sul lavoro del redattore.

L'articolo porterà avanti due finalità: 1) l'analisi di alcuni casi in cui un autore o un word processor mettono alla prova il lavoro del redattore (di alcuni casi spiegheremo la semantica o le cause, chiarendo ulteriormente la natura degli errori commessi) e 2) la scrittura di uno script bash che scovi per noi tutti i casi esplicitati nell'articolo e potenzialmente dannosi nei file LaTeX, nativi o convertiti dagli originali word processor. A causa della mia predilezione per i sistemi Unix-like la seconda finalità sarà principalmente indirizzata agli utenti di tali sistemi (come Linux e Mac OS X), ma non disdegna gli utenti di Windows che abbiano installato un interprete di comandi bash sul proprio computer (Cygnus Solution-Red Hat (2019) oppure Bruessow (2017) tra le soluzioni più note). Lo script avrà il solo compito di analisi perché in alcuni casi, discussi puntualmente nell'articolo, la correzione automatica non farebbe che aumentare i problemi.

L'articolo prosegue così: la sezione 2 presenta brevemente la struttura dell'algoritmo[3] alla base

[1] La percentuale attuale è circa il 2 % di manoscritti LaTeX e 0 % di manoscritti LyX e Libre/OpenOffice. Serve specificare quale programma è usato nel 98 % dei casi?

[2] Fortunatamente esistono i correttori ortografici che possono aiutare su questo fronte. Qualche word processor si spinge al controllo semantico...

[3] Ricordiamo che i termini algoritmo e procedura non sono equivalenti. L'algoritmo (Bertossi, 1990, 11) «che significa procedimento [...] indica la descrizione precisa delle *azioni* che un *esecutore* deve compiere per giungere alla soluzione di qualsiasi *problema* computazionale. [...] [P]uò essere considerato come un manipolatore di dati che, a fronte di certi dati d'ingresso consistenti con la natura del problema da risolvere (dati di *input*), produce altri dati come risultato

dello script descrivendone le parti più significative del codice; le sezioni 3–6 discuteranno alcuni casi d'uso reali tra quelli che ricadono nei punti 1–4 del precedente elenco e il relativo codice bash. Poiché alcuni di questi casi sono significativi per la sola lingua italiana, adattare lo script per altre lingue comporterà modificare alcuni test oppure cancellarne alcuni per aggiungerne di significativi nella lingua prescelta. Prima di concludere, la sezione 7 presenterà per intero lo script, ne descriverà alcuni dettagli implementativi e lo metterà all'opera su alcuni file di prova per testarne il funzionamento e mostrarne i risultati.

2 Impostazione dello script

Sebbene niente possa (ancora) sostituire l'accurato controllo umano di un testo scritto da un umano, un aiuto automatico nei compiti più ripetitivi è sempre benvenuto. Uno dei compiti più ripetitivi che si possa immaginare è controllare se tutte le voci di una *checklist* siano presenti o meno in un documento, specie se questo è costituito da più di qualche file (o di qualche pagina se siamo di quelli che lavorano coi fogli stampati). Ritengo che a qualunque redattore possa far comodo avere un programma che controlli la *checklist* in sua vece e gli faccia un rapporto su quali file contengano quali voci. La presenza di un rapporto riduce e circoscrive i controlli manuali e, soprattutto, funge da promemoria sulle eventuali correzioni da fare. Il programma avrà le sembianze di uno script bash che scriverà per noi il rapporto come conseguenza dell'analisi di uno o più file di testo (che ricordiamo essere il formato dei file `.tex`) alla ricerca dei casi discussi nelle prossime sezioni.[4]

Per prima cosa vogliamo che lo script usi bash indipendentemente dalla shell in uso.[5] Dunque la prima riga conterrà la sequenza di caratteri nota come *shebang*[6] e il comando completo di percorso assoluto:[7]

```
#!/bin/bash
```

Immediatamente dopo vogliamo controllare che l'analizzatore abbia ricevuto in input almeno un file:[8]

```
1  if␣[[␣$BASH_ARGC␣<␣1␣]];␣then
2  ␣␣echo␣"Uso:␣editanalyze␣<file\
3  ␣da␣analizzare>"
4  ␣␣echo␣"Es.:␣editanalyze␣*.tex\
5  ␣(controlla␣tutti␣i␣file␣con\
6  ␣estensione␣.tex)"
7  ␣␣echo␣"␣␣␣␣␣editanalyze\
8  ␣capitolo1.tex␣(controlla\
9  ␣il␣solo␣file␣capitolo1.tex)"
10 ␣␣echo␣"␣␣␣␣␣editanalyze\
11 ␣capitolo[1-5].tex␣(controlla\
12 ␣i␣file␣capitolo1-capitolo5\
13 .tex)"
14 ␣␣exit␣1
15 fi
```

Questo pezzo di codice controlla che il comando abbia ricevuto almeno un argomento (riga 1; `$BASH_ARGC` vale 0 se non abbiamo dato argomenti al comando); in caso negativo, stampa alcune righe riassuntive sull'uso dello script, mostrando esplicitamente la possibilità di ricorrere a *wildcard* e a espressioni regolari (Goyvaerts, 2019) (righe 2–13), quindi interrompe l'esecuzione dello script uscendo con codice di errore 1 (riga 14). Se il test precedente è negativo (cioè `$BASH_ARGC` $\geqslant 1$) abbiamo passato almeno un parametro e quindi lo script può continuare.

Vogliamo che il rapporto, memorizzato nel file `report.txt`, contenga l'elenco dei file in cui è stato riscontrato ognuno dei casi oggetto di analisi, elenco preceduto dall'indicazione del relativo caso, come per esempio:

```
Il carattere ° si trova in
1.tex
3.tex
```

Capiamo quindi che i casi da esaminare si riducono alla presenza o meno di uno o più caratteri (una stringa) in configurazioni più o meno lineari. Prima di vedere l'algoritmo, diamo due definizioni di comodo:

Definizione 1. Un file è *positivo* (all'analisi) se contiene almeno un'occorrenza del testo cercato.

Definizione 2. Un file è *negativo* (all'analisi) se non contiene neanche un'occorrenza del testo cercato.

del problema (dati di *output*)». «Un algoritmo descritto per mezzo di *costrutti* tipici di un linguaggio di programmazione è comunemente detto *procedura*» (Bertossi, 1990, 13).

[4] L'idea di produrre un tale script è nata successivamente all'idea di un articolo che descrivesse gli errori più frequenti degli autori, argomento tutto sommato abbastanza circoscritto. La versione dello script qui presentata è embrionale, forse didattica e quasi per niente testata in casi reali. In una versione precedente dell'articolo il redattore ha giustamente evidenziato l'incapacità dello script di trattare nomi di file contenenti degli spazî. Tutte le altre mancanze dello script, dal controllo sull'esistenza dei file di input alla scelta arbitraria del nome del file contenente il rapporto passando per la suggerita inefficienza, verranno implementate in futuro.

[5] Alcuni sistemi Unix usano, di default o per scelta amministrativa, shell diverse da bash, quali sh, [t]csh, ksh e molte altre.

[6] Così viene chiamata, per esempio, su Powers *et al.* (2002). Altre fonti, quali Wikipedia (2019) ne danno altre versioni e possibili significati.

[7] Bisogna comunque ricordarsi di rendere lo script eseguibile, altrimenti il sistema operativo non lo riconoscerà come comando e non lo eseguirà.

[8] Da notare che i testi dentro le virgolette dopo ogni comando `echo` vanno scritti su una riga. Nel caso la riga sia troppo lunga, potremo interromperla col carattere *di fuga* (comunemente detto *di escape*) `\`. Facciamo attenzione al numero di spazî dopo di esso per mostrare sullo schermo il testo di aiuto pulito e ordinato. In tutti i successivi brani di codice eviteremo l'uso esplicito del carattere di fuga e lasceremo a LaTeX l'incombenza di mandare a capo (solo tipograficamente) le righe troppo lunghe.

L'algoritmo proposto è il seguente:

```
stampa sullo schermo il caso da analizzare
per ognuno dei file ricevuti in input
  il file è positivo?
    SÌ: stampa il caso nel rapporto
        stampa il nome del file nel rapporto
        esci dal ciclo
    NO: non fare niente
per ognuno dei rimanenti file da analizzare
  il file è positivo?
    SÌ: stampa il nome del file nel rapporto
    NO: non fare niente
```

Il primo ciclo assicura che il caso verrà stampato nel rapporto solo se c'è un file positivo, il cui nome verrà inserito nel rapporto. Il secondo ciclo controlla i rimanenti file e stampa nel rapporto i soli positivi. Scritto in *bash-like*, l'algoritmo si presenta così:

```
1  echo␣"ANALISI␣DEL␣CASO␣IN␣ESAME"
2  count=0
3  for␣i␣in␣"$@";␣do
4  ␣␣if␣(␣CONDIZIONE␣DI␣TEST␣);␣then
5  ␣␣␣␣echo␣"TESTO␣DEL␣CASO␣IN␣ESAME"␣
       >>␣report.txt
6  ␣␣␣␣echo␣"$i"␣>>␣report.txt
7  ␣␣␣␣break
8  ␣␣fi
9  ␣␣count=$[$count+1]
10 done
11 args=("$@")
12 for␣((␣count=$[$count+1];␣$count<
       $BASH_ARGC;␣count=$[$count+1]␣))
       ;␣do
13 ␣␣CONDIZIONE␣DI␣TEST
14 done
```

Le locuzioni "condizione di test", "analisi del caso in esame" e "testo del caso in esame" sono in linguaggio naturale e andranno sostituite con del codice o delle scritte significative che vedremo nelle prossime sezioni. Soffermiamoci brevemente sul significato del codice scritto finora.

Le righe 1, 5 e 6 stampano qualcosa; a video se l'ultimo carattere della riga è ", su file in modalità *append* se dopo le virgolette di chiusura troviamo la sequenza >> seguita dal nome di un file. La riga 3 è un ciclo che assegna alla variabile i il contenuto di "$@"[9]. Quest'ultimo, stando a Ramey e Fox (2010, p. 24), «è un parametro speciale che si espande nei parametri posizionali partendo da uno. Quando l'espansione avviene entro i doppî apici, ogni parametro si espande in parole separate. Cioè, "$@" è equivalente a "$1", "$2"...». Dunque "$@" è un vettore contenente i parametri passati allo script e i assumerà il contenuto di ognuna delle celle di detto vettore, cioè i nomi dei file da analizzare.

La riga 4, così come la riga 13, deve testare una condizione. Nelle prossime sezioni espliciteremo una gamma di esse.

Per fare in modo che il secondo ciclo inizi dal primo file non ancora analizzato sfruttiamo il contatore (count) inizializzato a 0 nella riga 2 e incrementato di 1 ogni volta che troviamo un file negativo nel primo ciclo. Nel secondo ciclo dobbiamo iniziare a valutare dal file successivo all'unico positivo trovato. Purtroppo non è possibile scandire $@ indicizzandolo come un array, quindi dovremo assegnarne il contenuto a un array (che chiameremo args e che sappiamo partire dall'elemento 0, non da 1 come $@) e scandire quest'ultimo dalla posizione successiva a count fino alla fine.

I lettori più attenti saranno già insorti: se la riga 13 deve verificare una condizione di test per decidere cosa fare dopo, dove sono l'if e il then presenti nell'analoga riga 4? Risponderemo a questa domanda nella prossima sezione.

3 Caso 1: glifi errati

3.1 Analisi del caso

Il caso dei glifi errati è un caso in cui non sempre è possibile procedere a una correzione indiscriminata perché spesso non possiamo sapere se tale glifo è errato o no se non analizzando il contesto.

Un caso esemplare è l'uso del simbolo dei gradi (°) al posto della 'o' soprasegnata (º).[10] Agli occhi dei non amanti della tipografia i due simboli possono sembrare uguali, ma non lo sono e veicolano due significati diversi: la 'o' soprasegnata indica che il numerale immediatamente precedente dev'essere letto non come cardinale (uno, due, tre...) ma come ordinale maschile (primo, secondo, terzo...); il simbolo dei gradi va letto come "grado" o "gradi" se preceduto rispettivamente da 1 o dagli altri numeri e indica un angolo (normalmente espresso in gradi sessagesimali: un angolo giro vale 360°) o una temperatura se diversamente indicato.[11] In entrambi i casi (o soprasegnata e simbolo di gradi angolari) il simbolo va attaccato al numero che lo precede. Tipograficamente una 'o' soprasegnata mantiene le caratteristiche di forma, spessore e orientazione della 'o' del font in uso (alcuni font

[9] Le virgolette servono a salvare la situazione nel caso il nome del file contenga degli spazî: senza le virgolette un unico nome contenente spazî verrebbe suddiviso in una sequenza di tanti nomi di file quanti sono gli spazî più uno. Nel caso questi "nomi" non corrispondano a niente avremo dell'output di grep che ci avverte dell'inesistenza del file. Ma se i nomi errati corrispondono a qualche file, l'analizzatore scandaglierà dei file che forse non erano da analizzare.

[10] Per amor di *par condicio*, di recente mi è capitato di rileggere un libro (Polidoro, 2012) in cui la o soprasegnata è usata per indicare i gradi!

[11] Per indicare una temperatura, il simbolo ° va seguito da una lettera senza alcuno spazio tra i due glifi: C a indicare i gradi Celsius (scala in cui 0 indica la temperatura di congelamento dell'acqua e 100 quella di ebollizione), F per i gradi Fahrenheit (in cui le due temperature precedenti sono, rispettivamente, 32 e 212). Inoltre il simbolo va staccato con uno spazio breve insecabile (\,) dal numero che lo precede. Attenzione a un errore comune: la temperatura indicata in gradi Kelvin usa il solo simbolo K, senza °.

le aggiungono una sottolineatura, ma non tutti, e comunque detta sottolineatura non ci sarà mai se usiamo il comando `\textsuperscript{o}` invece di `\textordmasculine`) mentre il simbolo dei gradi ha spessore uniforme e nessuna orientazione (di fatto è una piccola circonferenza; normalmente i manuali come Oetiker *et al.* (2018) consigliano l'uso del comando matematico `\circ` all'esponente o di `\textdegree` del pacchetto textcomp. Notiamo che possiamo usare il simbolo di gradi presente sulla tastiera solo usando textcomp; diversamente avremo un errore in compilazione con pdfLaTeX). Nella tabella 1 vediamo i glifi ingranditi per meglio apprezzarne le differenze. Mi si potrebbe obiettare che le 'o' dei font Sans Serif sono più facilmente confondibili, ma non necessariamente queste sono disegnate come circonferenze o sono di spessore uniforme (cose che avvengono, per esempio, nel font Futura).

Tabella 1: A sinistra la 'o' soprasegnata (tra parentesi quella ottenuta con comando di soprascrittura); a destra il simbolo dei gradi (tra parentesi quello ottenuto col comando matematico e quello col comando testuale, identico al simbolo di riferimento).

$\underline{\text{o}}$ $\left({}^{\text{o}}\right)$	$^{\circ}$ $\left({}^{\circ}, {}^{\circ}\right)$

Molti autori, però, ignorano o fingono di ignorare le differenze elencate e mostrate. Pertanto trovano comodo o lecito usare un simbolo presente sulla totalità delle tastiere. Tale simbolo *sembra* proprio essere quello desiderato e non occorre dover impazzire a cercare quello corretto all'interno di sterminate mappe di caratteri.

Entrambi i simboli si trovano alla destra di numeri cardinali (scritti in cifre) e, dunque, solo la lettura del testo ci permette di capire se l'autore intendeva scrivere le cifre in gradi o le abbreviazioni di numeri ordinali. Quindi non è possibile applicare una sostituzione indiscriminata senza la certezza che tutti i casi ricadano in una e una sola delle due possibilità.

3.2 Codice di analisi

È arrivato il momento di svelare il "mistero" dei test introdotto alla pagina 3. Iniziamo proprio dalla ricerca del carattere °. All'interno dell'`if` della riga 4 scriveremo:

```
grep␣--silent␣-E␣°␣"$i"
```

quindi l'intera riga 4 sarà:

```
if␣(␣grep␣--silent␣-E␣°␣"$i"␣);␣then
```

Vediamone il significato, che potrebbe risultare oscuro a qualche lettore, a partire dalle singole componenti:
`if`: comando di bash che esegue un test di verità/falsità. Esegue i comandi seguenti la clausola `then` solo nel caso in cui il test nelle parentesi sia vero o valga 0;
`grep` (Magloire *et al.*, 2017): comando che stampa le righe di un file contenenti una stringa o pattern di riferimento (in pratica cerca la presenza di uno o più termini o di un'espressione regolare in uno o più file). Se tale stringa di riferimento è presente, lo stato di uscita di `grep` sarà 0. L'opzione `--silent` sopprime il normale output del programma (cioè la riga contenente la stringa di riferimento, eventualmente preceduta dal nome del file, che non vogliamo nell'output dello script) e l'eventuale codice 2 emesso in caso di errore (maggiori dettagli su Magloire *et al.* (2017, 12)) mentre l'opzione `-E` permette le espressioni regolari estese. Friedl (2006) dedica diverse pagine alle espressioni regolari di `egrep`. Questo comando è ormai deprecato e sostituito proprio da `grep -E`;
`°`: stringa o pattern di riferimento, in questo caso costituita da un solo carattere;
`"$i"`: contenuto della variabile `i` dentro cui (riga 2 del listato alla pagina 3) troviamo i nomi dei file passati come parametri al comando di analisi.

Il "test" da scrivere nella riga 13, invece, non prevede l'uso di `if` perché procederemo diversamente, usando l'espressività del sistema Unix e dei suoi comandi:

```
grep␣-l␣-E␣°␣"${args[$count]}"␣>>␣report.txt
```

Prima di analizzare per intero il "test", vediamo il significato delle singole parti:
`grep -l -E °`: di `grep` e `-E` già sappiamo; `-l` è l'opzione che sopprime il normale output di `grep` per stampare il nome dei file di input in cui sia stata trovata un'occorrenza della stringa di riferimento;
`"${args[$count]}"`: contenuto della cella `count`-esima dell'array `args` (ne sappiamo lo scopo dalla fine della sezione 2);
`>> report.txt`: ridirezione dell'output su file (qui chiamato `report.txt`) in modalità *append* (con scrittura di seguito a quanto già presente nel file; differisce dalla modalità *write* perché quest'ultima sovrascrive qualunque contenuto).

Alla luce di quanto appena visto, il "test" significa: stampa i nomi di tutti e soli i file specificati in input in cui trovi un'occorrenza della stringa ° e scrivili alla fine del file `report.txt`. Naturalmente non ci sarà alcuna scrittura nel rapporto se `grep` non trova alcuna stringa di riferimento in un file.

Ora abbiamo tutti (o quasi) gli elementi per arricchire lo script di tutti i controlli che riterremo necessarî.

3.3 Altri glifi errati

Due casi in cui l'errore è imputabile più al word processor che all'autore coinvolgono le virgolette "intelligenti". Il primo si ha quando scriviamo la forma abbreviata di un decennio, ad esempio *gli anni '20*. I normali word processor, vedendo che l'apostrofo è stato digitato dopo uno spazio, ritengono trattarsi di una virgoletta aperta e cambiano glifo, col risultato di avere *gli anni '20*. Il secondo caso si ha, invece, quando a una parola terminante con l'apostrofo succede una parola virgolettata, ad esempio *l'"attor giovane"*. Qui il word processor fa il "ragionamento" inverso: le virgolette sono state digitate senza avere uno spazio precedente, quindi devono essere virgolette chiuse. Dunque avremo ottenuto la stringa *l'"attor giovane"*. Ovviamente non possiamo sapere a priori se ogni eventuale occorrenza ricade in quanto già esposto o nei possibili "virgoletta aperta + numero cardinale" e "virgoletta chiusa + virgolette chiuse". Un controllo manuale dirimerà la questione.

Trovare questi casi è semplice: basta sostituire nel test il carattere `°` con la stringa `‘[0-9]` o con `’’’`. Insomma, sono tutte variazioni sul tema per cui funzionano anche le espressioni regolari.

Altri casi che potremmo voler controllare sono:

- spazî precedenti le interpunzioni. La complessità dell'analisi deriva dal fatto che gli autori possono aver compreso gli spazî in uno stile e nel risultante `.tex` dovremo tener conto delle parentesi graffe: `[␣]}*[␣]*[.,:;]`;

- spazî forzati (`\\\␣` è la stringa di riferimento da dare a `grep`), che il convertitore di un comune word processor inserisce anche nei casi in cui siano stati digitati più spazî consecutivi. Ciò inficia la proprietà di LaTeX di considerare più spazî consecutivi come un singolo spazio;

- trattini (dash, en-dash, em-dash). Qui la stringa di riferimento è più complicata perché gli autori tendono a "mescolare una gran quantità di... stili". Diciamo che può essere utile iniziare il controllo da `[A-Za-z.,:;␣]-[A-Za-z.,:;␣]` e le sue varianti con en-dash (codice Unicode 2013) e em-dash (codice Unicode 2014). Naturalmente il compito del redattore è verificare che ogni trattino analizzato sia stato usato secondo le norme redazionali.

Lascio al lettore trovare altri casi simili appartenenti alla stessa classe di problemi.

4 Caso 2: scorretta applicazione delle proprietà al testo

Spesso vedo, o mi vengono segnalate dagli editor che rivedono i miei impaginati, porzioni di testo di stile incoerente. Gli utenti di word processor hanno la comodità di poter selezionare una o più parole a cui applicare il grassetto, il corsivo e il sottolineato evidenziandole col mouse e poi premendo un pulsante (il maiuscoletto e l'inclinato sono normalmente di applicazione più farraginosa). Bene!, proprio questa comodità sembra incrementare la disattenzione: trovo sovente parole per metà in corsivo[12] e per metà in tondo e interpunzioni comprese nel corsivo quando dovrebbero essere in tondo. Come possiamo far esaminare questi due casi a `grep`?

Il primo caso, indice di trascuratezza, può essere esaminato abbastanza facilmente cercando quelle occorrenze di testo in cui qualche carattere precede o segue un comando `\text..`[13] senza alcuna interruzione. Questo è un caso (al pari di quelli discussi nella sezione 3.3) in cui l'uso delle espressioni regolari permette sintesi espressiva ed efficienza programmativa. Le stringhe da ricercare sono, rispettivamente:[14]

```
[0-9A-Za-z]\\\text(it|bf|sc|tt|sl)
```

e

```
\\\text(it|bf|sc|tt|sl){[^}]*}[0-9A-Za
    -z]
```

Il secondo caso è abbastanza simile: dobbiamo controllare la presenza di parti di testo con comandi `\text..` contenenti del testo che inizia e/o finisce con un segno di interpunzione. All'interno del `grep` possiamo scrivere le seguenti espressioni regolari entro una coppia di virgolette:

```
\\\text(it|bf|sc|tt|sl){[^}]*[␣
    ]*[.,:;][␣]*}
```

e

```
\\\text(it|bf|sc|tt|sl){[.,:;][␣
    ]*[^}]*}
```

rispettivamente per le interpunzioni alla fine e all'inizio di un `\text..`

Un terzo caso, non rilevabile da un redattore umano che guardi solo il PDF perché non porta conseguenze visibili ai documenti finali, riguarda sempre la scorretta applicazione delle proprietà al testo: una stringa viene resa in uno stile o peso in due o più riprese. Questo si traduce in due o più comandi `\text..` consecutivi, intercalati o no da spazî. Perché dovremmo voler rilevare questi casi? Perché nell'eventuale ebook prodotto a partire da quei sorgenti viene inserito uno spazio in corrispondenza dei due `\text..` consecutivi (anche non intervallati da uno spazio) e può capitare che detto spazio divida una parola, introducendo un refuso. Quindi sarà utile cercare la seguente stringa:

```
\\\text(it|bf|sc|tt|sl){[^}]*}[␣]*\\\
    text(it|bf|sc|tt|sl)
```

[12] L'uso del termine corsivo è di comodo. Quanto detto vale per il grassetto e le altre forme elencate.

[13] In questo caso il `.` è una *wild card* che indica *qualunque carattere* (dovrei specificare in questa sede "tra quelli leciti").

[14] Ho evitato di comprendere il sottolineato, il cui comando è `\underline`, per questione di inopportunità tipografica.

5 Caso 3: mancanza di stile

Raramente mi sono capitati manoscritti in cui gli autori abbiano usato gli stili. Nel linguaggio degli elaboratori di testo (ma anche degli editor HTML), gli stili sono delle proprietà visuali e strutturali da applicare a una porzione di testo. Per esempio, Intestazione 1, Titolo, Citazione, sono stili reperibili nel menù Stili di LibreOffice Writer e sono gli equivalenti di `\section`,[15] `\title`, `\begin{quote} \end{quote}` di LaTeX.

Poiché gli autori sono spesso restii a strutturare il loro testo, l'elemento da ricercare (meglio, di cui ricercare l'assenza) è proprio la traccia della struttura. Dunque in questo caso dovremo fare una ricerca "negata". Ad esempio, visto che normalmente un manoscritto convertito da un word processor sarà convertito come articolo, ci basterà cercare la presenza (o meglio l'assenza) di `\section`.

Come modifichiamo l'`if` della riga 4 per trovare il primo file in cui non sia presente il comando `\section`? È semplice: se il test visto in precedenza doveva testare una condizione di verità (la presenza di un simbolo), ci basta negare quella condizione per testare la falsità (l'assenza di un simbolo o di una stringa di testo):

```
if␣(␣!␣grep␣--silent␣-E␣\\\section␣$i␣
    );␣then
```

La negazione si esplicita col punto esclamativo (`!`).

Per il test successivo, quello della riga 13, sfruttiamo un flag di `grep` diverso da `-l` usato finora: `-L`. Questo flag permette a `grep` di elencare tutti e soli i file, tra quelli analizzati, che *non* contengono la stringa di riferimento:

```
grep␣-L␣-E␣\\\section␣${args[$count]}␣
    >>␣report.txt
```

Certamente un file non strutturato darà più lavoro al redattore rispetto a un file già strutturato anche se quest'ultimo, proprio in virtù di una strutturazione arbitraria da parte del convertitore o incompleta da parte dell'autore, non lo esenterà da un lavoro di "promozione" dei comandi di struttura (`\section`→`\chapter`, `\subsection`→`\section` e così via. Mi raccomando: non nell'ordine inverso. Perché? La risposta alla fine dell'articolo.) o dall'applicazione manuale del testo citato o altro.

6 Caso 4: documenti fintamente plurilingue

Mi capita spesso che i documenti convertiti da un word processor a LaTeX contengano una quantità da enorme a spropositata di `\selectlanguage` e `\foreignlanguage` (col termine "spropositata" intendo decine di `\selectlanguage` e centinaia di `\foreignlanguage` in manoscritti di non più di 50 cartelle). La presenza del primo è giustificabile senz'altro all'inizio del documento per impostare la lingua principale e, sporadicamente, all'interno di un documento in caso ci siano lunghe porzioni di testo di cui specificare la lingua. Un uso indiscriminato, spesso paragrafo per paragrafo nel caso di conversione automatica, non è giustificabile. La presenza del secondo comando, `\foreignlanguage`, è giustificabile in tutti i casi di porzioni di testo, possibilmente brevi, scritte in una lingua diversa da quella principale.

Una presenza spropositata dei due comandi in un manoscritto convertito da un word processor può indicare che l'autore non abbia etichettato correttamente il testo in base alla lingua, o che abbia lasciato la lingua preimpostata per redigere un manoscritto in un'altra lingua. Naturalmente il convertitore non può conoscere le intenzioni dell'autore e dunque si limiterà a porre quei comandi nei punti esatti in cui ne ravvisa la necessità. Facciamo un esempio. Supponiamo che l'autore abbia scritto il testo della figura 1 evidenziandolo allo stesso modo. Quando selezionerà l'inglese, questo verrà applicato a tutto il titolo evidenziato.

Allo stesso modo di come molti autori mettono i grassetti e i corsivi, anche mettere le lingue col mouse comporta degli inconvenienti. Volendo applicare l'inglese allo stesso testo, ma evidenziato come nella figura 2 comporterà impostarlo per ognuno dei due pezzi evidenziati; il risultato della conversione sarà avere due `\foreignlanguage` consecutivi anziché uno.

Quello appena discusso sarebbe un caso già ottimale o subottimale. Purtroppo la maggior parte delle volte gli esiti delle conversioni sono ben peggiori, con lingue impostate con qualche criterio oscuro e sbagliate (ignoro il motivo per cui trovai impostato il polacco per contrassegnare dei titoli di libri in francese e inglese...)

Comunque, basta fare un paio di ricerche sulle sole stringhe `selectlanguage` e `foreignlanguage`. Non è detto che la presenza di tali comandi in un documento sia indice di errore e quindi starà al redattore analizzare se e cosa correggere.

7 Il test finale

L'algoritmo bash-like così come scritto alla pagina 3 non è granché utile se tradotto pedissequamente nel programma: andrebbe ripetuto e adattato per ognuno dei casi da analizzare. Ciò sarebbe uno spreco di memoria (limitato, ma spreco), di tempo di digitazione (pure al netto del copia e incolla) e, soprattutto, di manutenzione (immaginate di aver scritto male un test dal punto di vista semantico *prima* del copia e incolla; lo script è stato fatto per analizzare centinaia di casi e dovete correggere il test per tutte le centinaia di casi). Possiamo trovare un automatismo che ci permetta di usare lo stesso codice per tutti i casi simili, un po' come le

[15] Sembra che il convertitore di Writer non contempli altro tipo di documento che non sia un articolo, da lì l'equivalenza Intestazione 1-`\section`.

… nel loro fondamentale *A Programming Approach to Computability* gli autori…

Figura 1: L'autore selezionerà l'inglese per il testo evidenziato.

… nel loro fondamentale *A Programming* *Approach to Computability* gli autori…

Figura 2: L'autore dovrà selezionare l'inglese per ognuna delle porzioni di testo evidenziate.

funzioni del linguaggio C? Sì, esiste: basta sfruttare gli array come già fatto per i parametri di input.

Le figure 3 e 4 mostrano l'intero codice dello script di analisi (`editanalyze`), suddiviso per motivi di spazio ma in maniera significativa: una figura contiene la parte "dichiarativa", l'altra quella "imperativa".[16] Nel codice mostrato nella figura 3 riempiamo una per una le celle di sei array. Questi sei vettori contengono il pattern di ricerca (`stringa`), il testo di output a video (`caso`) e il testo da scrivere nel rapporto (`testo`) per il test di positività e gli analoghi per il test di negatività (`stringan`, `cason`, `teston`). Tali array, espandibili man mano che ci si presentano nuovi casi da voler includere nell'analisi, ci permettono di parametrizzare gli elementi variabili nel codice, che non dovremo più duplicare per ogni caso da analizzare. Per ognuno degli array scriveremo una variabile col valore delle celle riempite, così da non dover cambiare valori all'interno del ciclo che testa in sequenza tutti i casi noti (trattiamo una variabile come fosse una costante).

Sempre nella figura 4, dopo il test sugli argomenti, notiamo che viene cancellata un'eventuale vecchia versione di `report.txt`, quindi ci sono due blocchi analitici: quello per tutti i casi di positività rispetto a un pattern e quello per verificare l'unico caso studiato in cui ci interessa che il pattern sia assente. Avremmo potuto accorpare in un unico codice i due casi? In definitiva si tratta di aggiungere o togliere un punto esclamativo a un test e sostituire un `-l` con un `-L` o viceversa. La cosa sarebbe fattibile in diversi modi: mi viene in mente di usare `sed` e su Unix & Linux Stack Exchange suggeriscono, in aggiunta, di usare uno script Perl per modificare quanto necessario o di sostituire lo script con uno script esterno. Ma perché vogliamo complicarci la vita, complicando anche quella del sistema operativo (quelli moderni sono progettati per evitare programmi con codice automodificante perché fonte di potenziali problemi di sicurezza e di determinismo esecutivo oltre che, aggiungerei, di leggibilità del codice), solo per risparmiare 1 KB di codice e provare a imitare i puntatori a funzione del C? Personalmente non mi avventurerò per questa via.

Passiamo al test. Abbiamo costruito quattro file contenenti ognuno alcuni casi tra quelli elencati nell'articolo. Il loro contenuto è mostrato nella tabella 2.

La figura 5 riporta il contenuto del rapporto generato dall'esecuzione dello script sui quattro file (`editanalyzer *tex`).

8 Conclusioni

Il lavoro del redattore può essere molto pesante quando arriva il momento di controllare tutte quelle piccole minuzie relative alle più elementari regole tipografiche. Uno strumento automatico di analisi delle minuzie può essere di grande aiuto, sia dal punto di vista dell'esaustività, sia da quello della velocità.

Questo articolo ha voluto fornire il suo contributo alla spiegazione degli errori commessi dagli autori, chiarendone la semantica e i motivi quando opportuno, e alla costruzione di uno strumento automatico di controllo.

Quest'ultimo punto si concretizza in uno script bash che permette di controllare uno o più file di testo alla ricerca di una serie di errori tipici e che genera un rapporto descrittivo della corrispondenza file-caso analizzato. Starà poi al redattore determinare quali occorrenze dovranno essere corrette e come correggerle.

Infine, questo lavoro potrebbe essere considerato un primo passo verso il controllo esaustivo automatico della corretta applicazione delle norme tipografiche in un manoscritto.

Ringraziamenti

L'obbligatorio ringraziamento va al Consiglio Scientifico del GuIT per aver valutato positivamente il lavoro e per i consigli datimi. Ma il ringraziamento grande va all'ottimo redattore che mi ha fatto notare alcune imprecisioni, dimenticanze e lungaggini nel testo e alcune dimenticanze nel codice del programma. La sua osservazione — in puro stile Pierre de Fermat — sulla dubbia efficienza dello script porterà in un prossimo futuro all'eventuale adozione di comandi alternativi a `grep` in base al risultato di test sulla velocità di esecuzione su una serie di testi di diverse dimensioni.

Risposta al quesito proposto alla pagina 6

Se iniziamo la promozione dal livello più basso, dunque sostituendo `\subparagraph` con `\paragraph`,

[16] Le virgolette ai termini dichiarativa e imperativa indicano un significato forzato. Bash, a differenza di linguaggi come il C, non distingue le due fasi e la suddivisione data qui è più formale che sostanziale.

```
#! /bin/bash

# Parte da modificare in base ai casi da analizzare
# Array dei pattern di ricerca "positiva" e dei messaggi per l’utente
stringa[1]="°"
caso[1]="Analisi della presenza del simbolo °..."
testo[1]="\nIl carattere ° si trova in"
stringa[2]="‘[0-9]"
caso[2]="Analisi della presenza dell’apostrofo sbagliato prima degli anni..."
testo[2]="\nL’apostrofo sbagliato prima degli anni si trova in"
stringa[3]="’’”"
caso[3]="Analisi della presenza della sequenza ’’”"
testo[3]="\nLa sequenza ’’’ si trova in"
stringa[4]="[ ]}*[ ]*[.,:;]"
caso[4]="Analisi della presenza di spazi prima delle interpunzioni..."
testo[4]="\nSpazi prima delle interpunzioni si trovano in"
stringa[5]="\\\ "
caso[5]="Analisi della presenza di spazi forzati..."
testo[5]="\nSpazi forzati si trovano in"
stringa[6]="[0-9A-Za-z.,:; ]-[0-9A-Za-z.,:; ]"
caso[6]="Analisi della presenza di trattini brevi..."
testo[6]="\nTrattini brevi si trovano in"
stringa[7]="[0-9A-Za-z.,:; ]-[0-9A-Za-z.,:; ]"
caso[7]="Analisi della presenza di trattini medi..."
testo[7]="\nTrattini medi si trovano in"
stringa[8]="[0-9A-Za-z.,:; ]-[0-9A-Za-z.,:; ]"
caso[8]="Analisi della presenza di trattini lunghi..."
testo[8]="\nTrattini lunghi si trovano in"
stringa[9]="[0-9A-Za-z]\\\text(it|bf|sc|tt|sl)"
caso[9]="Analisi della presenza di lettere o numeri prefissi a un comando \textxx..."
testo[9]="\nLettere o numeri prefissi a un comando \\\textxx si trovano in"
stringa[10]="\\\text(it|bf|sc|tt|sl){[^}]*}[0-9A-Za-z]"
caso[10]="Analisi della presenza di lettere o numeri postfissi a un comando \textxx..."
testo[10]="\nLettere o numeri postfissi a un comando \\\textxx si trovano in"
stringa[11]="\\\text(it|bf|sc|tt|sl){[^}]*[ ]*[.,:;][ ]*}"
caso[11]="Analisi della presenza di interpunzioni alla fine di un comando \textxx..."
testo[11]="\nInterpunzioni alla fine di un comando \\\textxx si trovano in"
stringa[12]="\\\text(it|bf|sc|tt|sl){[.,:;][ ]*[^}]*}"
caso[12]="Analisi della presenza di interpunzioni all’inizio di un comando \textxx..."
testo[12]="\nInterpunzioni all’inizio di un comando \\\textxx si trovano in"
stringa[13]="\text(it|bf|sc|tt|sl){[^}]*}[ ]*\\\text(it|bf|sc|tt|sl)"
caso[13]="Analisi della presenza di comandi \textxx consecutivi..."
testo[13]="\nComandi \\\textxx consecutivi si trovano in"
stringa[14]="selectlanguage"
caso[14]="Analisi della presenza di comandi \selectlanguage..."
testo[14]="\n\selectlanguage si trova in"
stringa[15]="foreignlanguage"
caso[15]="Analisi della presenza di comandi \foreignlanguage..."
testo[15]="\n\\\foreignlanguage si trova in"
casip=15

# Array dei pattern di ricerca "negativa" e dei messaggi per l’utente
stringan[1]="\\\section"
cason[1]="Analisi dell’assenza di \section..."
teston[1]="\nIl comando \section non si trova in"
casin=1
# Fine parte da modificare
```

Figura 3: Codice dello script di ausilio ai redattori.

```
if [[ $BASH_ARGC < 1 ]]; then
  echo "Uso: editanalyze <file\
 da analizzare>"
  echo "Es.: editanalyze *.tex\
 (controlla tutti i file con\
 estensione .tex)"
  echo "     editanalyze\
 capitolo1.tex (controlla\
 il solo file capitolo1.tex)"
  echo "     editanalyze\
 capitolo[1-5].tex (controlla\
 i file capitolo1-capitolo5.tex)"
  exit 1
fi

rm report.txt

# Analisi dei casi positivi (presenza di un pattern nei file)
for (( n=1 ; n<=$casip; n=n+1 )); do
  echo ${caso[$n]}
  echo "Pattern di ricerca: " ${stringa[$n]}
  count=0
  for i in "$@"; do
    if ( grep --silent -E "${stringa[$n]}" "$i" ); then
      echo -e ${testo[$n]} >> report.txt
      echo "$i" >> report.txt
      break
    fi
    count=$[$count+1]
  done
  args=("$@")
  for (( count=$[$count+1]; $count<$BASH_ARGC; count=$[$count+1] )); do
    grep -l -E "${stringa[$n]}" "${args[$count]}" >> report.txt
  done
done

# Analisi dei casi negativi (assenza di un pattern nei file)
for (( n=1 ; n<=$casin; n=n+1 )); do
  echo ${cason[$n]}
  echo "Pattern di ricerca: " ${stringan[$n]}
  count=0
  for i in "$@"; do
    if ( ! grep --silent -E "${stringan[$n]}" "$i" ); then
      echo -e ${teston[$n]} >> report.txt
      echo "$i" >> report.txt
      break
    fi
    count=$[$count+1]
  done
  args=("$@")
  for (( count=$[$count+1]; $count<$BASH_ARGC; count=$[$count+1] )); do
    grep -L -E "${stringan[$n]}" "${args[$count]}" >> report.txt
  done
done

exit 0
```

Figura 4: Seguito del codice dello script di ausilio ai redattori.

```
Il carattere ° si trova in
1.tex
3.tex

L’apostrofo sbagliato prima degli anni si trova in
2.tex
3.tex

La sequenza ’’’ si trova in
1.tex
3.tex

Spazi prima delle interpunzioni si trovano in
2.tex
3.tex

Spazi forzati si trovano in
1.tex
3.tex

Trattini brevi si trovano in
2.tex
3.tex

Trattini medi si trovano in
1.tex
3.tex

Trattini lunghi si trovano in
2.tex
3.tex

Lettere o numeri prefissi a un comando \textxx si trovano in
1.tex
4.tex

Lettere o numeri postfissi a un comando \textxx si trovano in
2.tex
4.tex

Interpunzioni alla fine di un comando \textxx si trovano in
1.tex
4.tex

Interpunzioni all’inizio di un comando \textxx si trovano in
2.tex
4.tex

Comandi \textxx consecutivi si trovano in
1.tex
4.tex

\selectlanguage si trova in
2.tex
4.tex

\foreignlanguage si trova in
1.tex
4.tex

Il comando \section non si trova in
1.tex
3.tex
```

Figura 5: Contenuto del rapporto sull’analisi dei quattro file riportati nella tabella 2.

Tabella 2: Contenuto dei quattro file testuali usati per provare lo script.

1.tex	2.tex
10°	‘20
,’’	abc ,
\	10-20
10-20	30-40
a\textit{abd}	\textit{abc}0
\textbf{abc,}	\textit{, abc}
\textsc{abc}\textsc{def}	\selectlanguage{italian}
\foreignlanguage{italian}{ciao}	\section{}
3.tex	**4.tex**
50°	T\textit{he fog}
‘70	\textit{Essi vivo}no
l’’’etica	\textsc{John Carpenter,}
\textit{allorquando },	\textsc{, John Carpenter}
sono \ qui	\textit{La} \textit{Cosa}
quando -travolti -	\selectlanguage{english}
quando - travolti -	\foreignlanguage{french}{aussi}
quando-travolti-	\section{}
Per qualche “oscuro” motivo, en-dash (–) e em-dash (—) vengono mostrati come dei normali dash (-) nel carattere monospaziato. Tecnicamente è chiaro che i tre diversi caratteri sono stati disegnati allo stesso modo e con le stesse dimensioni per rispettare la principale proprietà dei caratteri monospaziati, cioè l’uniforme dimensione orizzontale.	

come potremo distinguere i `\paragraph` originali da promuovere a `\subsubsection` da quelli appena promossi da `\subparagraph`?

Riferimenti bibliografici

Bertossi, Alan. A. (1990). *Strutture Algoritmi Complessità*. ECIG (Edizioni Culturali Internazionali Genova), Genova.

Bruessow, Christian V.J. (2017). «win-bash - bash port for windows». `http://win-bash.sourceforge.net/`.

Cygnus Solution-Red Hat (2019). «Cygwin». `http://www.cygwin.com/`.

Friedl, Jeffrey E.F. (2006). *Mastering Regular Expressions*. O’Reilly, Sebastopol, 3ª edizione.

Goyvaerts, Jan (2019). «Regular-Expression.info - Regex Tutorial, Examples and Reference-Regexp Patterns». `https://www.regular-expressions.info/`.

Magloire *et al.*, Alain (2017). *GNU Grep: Print lines matching a pattern*. `http://www.gnu.org/software/grep/manual/`.

Oetiker, Tobias, Hubert Partl, Irene Hyna e Elisabeth Schlegl (2018). *The Not So Short Introduction to LaTeX 2ε*. Accessibile da terminale con `texdoc lshort`.

Polidoro, Massimo (2012). *Il sesto senso*. Gruner+Jahr/Mondadori, Milano.

Powers, Shelley, Jerry Peek, Tim O’Reilly e Mike Loukides (2002). *Unix Power Tools*. Sebastopol, CA.

Ramey, Chet e Brian Fox (2010). *Bash Reference Manual*. Boston, MA.

Rawlinson, Graham Ernest (1976). *The Significance of Letter Position in Word Recognition*. Tesi di Dottorato, University of Nottingham.

Simons, Daniel J. e Christopher F. Chabris (1999). «Gorillas in our midst: Sustained inattentional blindness for dynamic events». *Perception*, **28** (9), pp. 1059–1074. `https://doi.org/10.1068/p281059`.

Simons, Daniel J. e Daniel T. Levin (1998). «Failure to detect changes to people during a real-world interaction». *Psychonomic Bulletin & Review*, **5** (4), pp. 644–649. `https://doi.org/10.3758/BF03208840`.

Wikipedia (2019). «Shebang (Unix)». `https://en.wikipedia.org/wiki/Shebang_(Unix)`.

▷ Gianluca Pignalberi
`g dot pignalberi at gmail dot com`

A Direct Bibliography Style for ArsTeXnica

Jean-Michel Hufflen

Abstract

We describe the `mlb-arstexnica` program, part of MlBibTeX's new version, and suitable for generating bibliographies for ArsTeXnica articles. First, we recall the notion of *direct* bibliography style related to MlBibTeX and mention the advantages of such a program. We show that our program provides additional services suitable for ArsTeXnica, compared to BibTeX's bibliography style arstexnica.bst.

Keywords BibTeX, MlBibTeX, LaTeX, biblatex package, Unicode, interface with Scheme.

Sommario

Si descrive il programma mlb-arstexnica, parte della nuova versione di MlBibTeX; esso è adatto per generare le bibliografie per gli articoli di ArsTeXnica. Si richiama la nozione di stile bibliografico *diretto* riferito a MlBibTeX e si sottolineano i vantaggi di questo programma. Si mostra che questo programma fornisce ulteriori funzionalità adatte ad ArsTeXnica in confronto a quanto si può ottenere con lo stile bibliografico artexnica.bst da usare con BibTeX.

Parole chiave BibTeX, MlBibTeX, LaTeX, biblatex, Unicode, interfaccia con Scheme.

1 Introduction

In some past GuIT conferences we have already introduced MlBibTeX[1], our implementation of a 'better' BibTeX (Patashnik, 1988b), the bibliography processor usually associated with LaTeX. Let us recall that a *bibliography processor* builds 'References' section—as source texts—from *citation keys* and *bibliography database* files. See Mittelbach and Goossens (2004, §§ 12.1.3 & 13.2) about LaTeX citation keys, extracted from *auxiliary* (.aux) files, and BibTeX's format of database (.bib) files. The BibTeX program is ageing, its bibliography styles are specified using an old-fashioned language based on handling a stack (Patashnik, 1988a). As mentioned in Mittelbach and Goossens (2004, § 13.6.3), introducing small changes within an existing style is quite easy, but designing new styles from scratch may be tedious. In addition, it hardly meets modern requirements such as dealing with formats extending the basic ascii[2] code, in particular, formats related to Unicode (e.g., utf-8[3]). Accented letters can be processed using TeX commands, but accent commands are ignored by BibTeX's sort procedure, so the lexical order provided by this program is only meaningful in English.

Nowadays more and more users typeset bibliographies for LaTeX documents with the biblatex package (Lehman, 2018), associated with the biber bibliography processor (Kime and Charette, 2018). These two tools[4] allow end-users to get access to many interesting extensions: for example, the fields `YEAR`, `MONTH` and `DAY`[5] can be replaced by the `DATE` field, also usable for date *ranges*, e.g., `2019-08-31/2019-09-06`. However the drawback of such extensions appears if users revert to 'old' BibTeX, since its standard styles do not recognise these extensions[6]. Sometimes, users have to do that, for example, if they put research articles onto some Web sites controlling the process of publishing in conference proceedings[7]. As another example, the bibliography style arstexnica.bst, used for the articles of the homonymous journal, is unable to deal with the extensions introduced by the biblatex package.

One year ago, we studied this bibliography style in order to fix a bug and thought that reimplementing it as a *direct style* of MlBibTeX could be useful for the ArsTeXnica board. In Section 2, we recall some general points about MlBibTeX, in particular the notion of direct style. Section 3 is a short comparison between BibTeX and MlBibTeX. The look of our proposed command is described in Section 4. Reading this article only requires basic knowledge of LaTeX and BibTeX.

2 MlBibTeX's Outlines

When we started MlBibTeX's development, we were mainly interested in multilingual aspects (Hufflen, 2005). Then we proposed some syntactical extensions in order to ease the specification of authors' and editors' names (Hufflen, 2006), we went thoroughly into some points re-

[1] **M**ulti**L**ingual BibTeX.

[2] **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.

[3] **U**nicode **T**ransformation **F**ormat.

[4] There are some descriptions of these tools in Italian: Pantieri (2009) for an introduction and Valbusa (2014) about advanced features.

[5] This last field does not belong to BibTeX standard, even if some styles use it.

[6] For example, the `YEAR` field is required if you use 'old' BibTeX and would like your bibliographies to be sorted; it cannot be replaced by the `DATE` field.

[7] The most famous site for Computer Science conferences is indisputably `http://www.easychair.org`.

```
%encoding = utf8

@BOOK{cussler2010,
      AUTHOR = {Clive Eric Cussler,
                abbr => Cl. with
                first => Jack,
                last => Du Brul},
      TITLE = {The Silent Sea},
      PUBLISHER = {Penguin Books},
      YEAR = 2010,
      LANGUAGE = english}

@BOOK{deturris1991,
      AUTHOR = {first => Gianfranco,
                last => De Turris},
      TITLE = {Il disagio della realtà},
      PUBLISHER = {Edizioni Settimo Sigillo},
      ADDRESS = {Roma},
      YEAR = 1991,
      LANGUAGE = italian}
```

Figure 1: Some syntactical extensions of MlBibTEX.

lated to programming, e.g., the definition of chaining ambitious language-dependent order relations (Hufflen, 2007) and enlarged expressive power by introducing *inexact* information about ancient documents (Hufflen, 2014). Since the first public version (Hufflen, 2003), MlBibTEX—written in Scheme—has been able to apply BibTEX bibliography styles or styles written using an extension of xslt[8] (W3C, 1999), the language used for transformations of xml[9] texts[10]. Then some existing styles have been wholly rewritten in Scheme, some new ones have been wholly designed in Scheme, too. Such styles—which are very efficient—are so-called *direct* with respect to MlBibTEX's terminology.

A new version, announced in Hufflen (2015), deals with Unicode and allows .bib files to use various encodings. If *several* .bib files are to be searched for document citation keys, *each* .bib file can use its own encoding. The program tries to guess the encoding used within such a file, but it is recommended to write this information down as we do in Fig. 1. The default encoding for input and output files is Latin 1, but can be changed within your initialisation files by means of the interface with Scheme. Fig. 1 shows some syntactical extensions provided by MlBibTEX.

Last but not least, let us recall that when MlBibTEX processes an .aux file, it also reads the preamble of the corresponding source .tex document[11]. What is important for our purposes is that MlBibTEX can detect the inputenc package option (Mittelbach and Goossens, 2004, § 7.1.2), that is, the encoding to be used for the output file containing generated references.

3 MlBibTEX vs BibTEX

If we consider some standard uses of bibliographical entries, the main difference between MlBibTEX and BibTEX is that the former is less permissive than the latter. Since its first version, MlBibTEX has performed more checks than 'old' BibTEX, and designing direct styles in Scheme allowed us to go on in this direction. For example, all the fields associated with a date must be well-formed: the `YEAR` field must be a non-zero integer[12], the `MONTH` field must be a mnemonic among `jan`, `feb`, . . . , `dec`. Likewise, the taxonomy of the values associated with the `DATE` field is checked. Some conventions about dates may appear as too drastic, but they insure that our chronological sort procedures work properly. Here are the other fields subject to a more advanced check than in BibTEX and usable in ArsTEXnica style:

- for *person names*, e.g., `AUTHOR` and `EDITOR`;
- for language names: `LANGUAGE`;
- for urls[13].

When a field name is unrecognised, a warning message is emitted: often this convention allows end-users to fix typing mistakes in practice. Here are the additional conventions when fields introduced by the biblatex package are used within bibliographical entries of .bib files but unrecognised within 'standard' bibliography styles:

- if the `DATE` field is used:
  - if it is associated with a single date, it is expanded using the fields `YEAR`, `MONTH` and `DAY`,
  - if it is associated with a range, the second date (the range's upper bound) is dropped out and the previous rules applies;
- to sort bibliographies, the fields `SORTYEAR` and `SORTTITLE`—when given—are used instead of `YEAR` and `TITLE`.

4 The `mlb-arstexnica` Program

There are two ways to process ArsTEXnica bibliographies with MlBibTEX:

- run the `mlbibtex` executable program and use the bibliography style arstexnica.bst;
- run the direct style `mlb-arstexnica`.

[8] e**X**tensible **S**tylesheet **L**anguage **T**ransformations.

[9] e**X**tensible **M**arkup **L**anguage.

[10] Parsing .bib files results in Scheme structures that may be viewed as xml trees, using an open format.

[11] On the contrary, 'old' BibTEX *never* reads .tex files, it only processes .aux files.

[12] . . . unless the `-inexact` option is used, in which case some digits may be replaced by '`?`'. See Hufflen (2014) for more details.

[13] **U**niform **R**esource **L**ocator.

The first way is still based on the .bst file, which may be viewed as more readable than a Scheme program. The second way results in a more efficient process and may get access to some operations unreachable by a .bst program: for example, using advanced or language-dependent order relations to sort bibliographies. The `mlb-arstexnica` executable file is added to the programs announced in Hufflen (2015, § 4). Its command line is:

```
mlb-arstexnica [option]* filename
```

filename being an .aux file; you can put the suffix or leave it implicit. Possible options are:

`-h` or `-help` displays help messages and exits;

`-inexact` allows *inexact* information to be accepted and processed: see Footnote 12, p. 2 and Hufflen (2014);

`-min-crossrefs=n` has the same effect than in BibTEX: entries accessed at least *n* times (*n* is a natural number) by means of a CROSSREF field are put; see Mittelbach and Goossens (2004, § 13.2.5) for more details;

`-tex-file=...` allows end-users to make precise the source LATEX file associated with the .aux file, when it cannot be easily deduced[14].

At the time of writing, MlBibTEX and its derived programs can run on Linux and Mac OS X; they should be able to run on Windows. We are in contact with the CTAN[15] in order to put our files onto this site. As most files available within a TEX distribution, our source files are subject to the LPPL[16].

5 Conclusion

We think that our `mlb-arstexnica` program can provide many additional services compared to the present style of BibTEX. We hope that end-users will play with it with as much pleasure as ours developping it.

Acknowledgements

I thank Claudio Beccari for his patience, and for his Italian translations of the abstract and keywords.

References

Hufflen, Jean-Michel (2003). «MlBibTEX's version 1.3». TUG*boat*, **24** (2), pp. 249–262.

— (2005). «MlBibTEX: a survey». In *Proc.* GUIT *Meeting*. Pisa, Italy, pp. 171–179.

— (2006). «Names in BibTEX and MlBibTEX». TUG*boat*, **27** (2), pp. 243–253. TUG 2006 proceedings, Marrakesh, Morocco.

— (2007). «Managing order relations in MlBibTEX». TUG*boat*, **29** (1), pp. 101–108. EuroBachoTEX 2007 proceedings.

— (2014). «Dealing with ancient works in bibliographies». *ArsTEXnica*, **18**, pp. 81–86. In Proc. GUIT meeting 2014.

— (2015). «MlBibTEX 1.4: the new version». *ArsTEXnica*, **20**, pp. 35–39. In Proc. GUIT meeting 2015.

Kime, Philip and François Charette (2018). *biber. A Backend Bibliography Processor for biblatex. Version biber 2.12 (biblatex 3.12).* http://ctan.org/pkg/biber.

Lehman, Philipp, with Philip Kime, Moritz Wemheuer, Audrey Boruvka and Joseph Wright (2018). *The biblatex Package. Programmable Bibliographies and Citations. Version 3.12.* http://ctan.org/pkg/biblatex.

Mittelbach, Frank and Michel Goossens, with Johannes Braams, David Carlisle, Chris A. Rowley, Christine Detig e Joachim Schrod (2004). *The LATEX Companion.* Addison-Wesley Publishing Company, Reading, Massachusetts, 2ª edizione.

Pantieri, Lorenzo (2009). «L'arte di gestire la bibliographia con biblatex». *ArsTEXnica*, **8**, pp. 48–60.

Patashnik, Oren (1988a). *Designing BibTEX Styles.* Part of the BibTEX distribution.

— (1988b). *BibTEXing.* Part of the BibTEX distribution.

Valbusa, Ivan (2014). «Funzionalità avanzate del sistema biblatex/biber». *ArsTEXnica*, **18**, pp. 70–80.

W3C (1999). XSL *Transformations (*XSLT*). Version 1.0.* http://www.w3.org/TR/1999/REC-xslt-19991116. W3C Recommendation. Edited by James Clark.

▷ Jean-Michel Hufflen
FEMTO-ST (UMR CNRS 6174) & University of Bourgogne Franche-Comté,
16, route de Gray,
25030 BESANÇON CEDEX
FRANCE
jmhuffle at femto-st dot fr

[14] Let us recall that MlBibTEX reads the source LATEX file's preamble (cf. § 2).

[15] **C**omprehensive TEX **A**rchive **N**etwork.

[16] LATEX **P**roject **P**ublic **L**icense. For more details, see https://www.latex-project.org/lppl.txt.

Smartdiagram: The Package and Its Journey

Claudio Fiandrino

Abstract

Smartdiagram born as a response to a question on TEX.stackexchange. The challenge was to emulate a feature that Microsoft Power Points provides: the capability of automate with animations a diagram. This feature allows to create diagrams from lists, so the user interface had to be as simple as possible, i.e., a list. In this article, I review the basic idea that overcomes the challenge and I expose the main features of the package along with a bit of its history.

Sommario

Smartdiagram è nato come risposta ad una domanda apparsa su TEX.stackexchange. La sfida proposta era emulare il comportamento di una funzionalità di Microsoft Power Points, ossia la capacità di creare in modo automatico un diagramma grafico da una lista di elementi con capacità di animazione dei singoli elementi. In questo articolo, si spiega l'idea di fondo in grado di vincere la sfida e si espongono le principali caratteristiche del pacchetto con una nota sull'evoluzione storica.

1 Introduction

Smartdiagram has born as a response to a question on TEX.stackexchange[1]. The challenge was to emulate a feature provided by Microsoft Power Points: the capability of creating a diagram from a list with animations. As LATEX does not share the What You See Is What You Get (WYSIWYG) paradigm, it is not possible to first write the list and then click to obtain the diagram. Nevertheless, to emulate the intuitive usage, the smartdiagram user interface transforming a list into the diagram had to be as simple as possible, i.e., characterizing intuitively the list, the type of diagram and the options to customize the diagram.

Based on TikZ Tantau (2010), the code employed in such answer has become the core of the smartdiagram package Fiandrino (2012). Along the subsequent months, many additional components have been included such as the support for the key-management interface, new types of diagrams and correction of (many) bugs and typos.

The package overcomes the initial challenge. However, despite its capability of full customization of the diagram provided by the key-management interface, smartdiagram limits somehow the user. Indeed, if one wants to move away from the predefined type of diagrams or do substantial modifications that break the automatism that creates the diagram, then the package is not suitable and it is better to resort to plain TikZ for such a job.

The remainder of the paper is structured as follows. Section 2 explains the steps that automatize the creation of the diagram. Section 3 details the operation of auxiliary components such as the key-management interface and the library to include additional elements to the diagram. Section 4 presents some examples and Section 5 briefly discuss the media coverage of the package. Finally, Sections 6 and 7 conclude the work and acknowledge contributors respectively.

2 The core idea

This section overviews the core idea that allows to automatize the creation of diagrams from lists. First of all, one, simple macro is in charge of such operation:

```
\smartdiagram[<type>]{<list of items>}
```

where `type` is the type of diagram (please refer to Fiandrino (2012) for a complete list) and `list of items` for the items that should appear in the diagram. For example:

```
\smartdiagram[sequence diagram]{%
Select type of diagram,
Count items,
Draw diagram
}
```

creates Fig. 1, which shows the essential building blocks that works both for the animated and non-animated modes. These two modes can be enforced with different macros, the above `\smartdiagram` and `\smartdiagramanimated`. The latter only works with the Beamer class. For the sake of the diagram composition, nothing changes between the two macros.

The first argument of `\smartdiagram` that defines the type of diagram undergoes a switch-case check. Then, following Fig. 1, the first operation that is performed commonly to all the types of diagrams is to count the number of items in the list. This allows to define for the specific diagram aspects like the angle of separation between the items for circular-like diagrams or the distance between blocks in a parametric fashion. By assigning to each item an ID, then it is possible to draw the interconnections.

[1] Available online at: `https://tex.stackexchange.com/q/78310/13304`

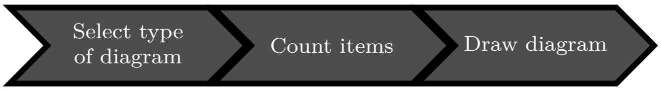


Figure 1: The building blocks of smartdiagram operation

The animated version of the diagram simply includes an overlay specification to the item that are progressively shown. The mechanisms that merge the capabilities of Ti*k*Z and Beamer is well described in Fiandrino (2014).

3 Auxiliary components

3.1 The key-management interface

Ti*k*Z allows to define custom key interfaces through its pgfkeys package. Smartdiagram resorts to this principle with a twofold purpose. On the one hand, to fully customize the aspect of the diagrams in terms of colors, dimensions of the boxes, the respective distance between the various elements of the diagram. On the other hand, like explained later, it allows to create separate libraries very easily.

In the following, this subsection will expose the main aspects of the smartdiagram key-management interface:

- All the keys related to smartdiagram go under the path `/smart diagram/`. This allows on the one hand to avoid conflicts with other packages, e.g., double names, and on the other hand to make easier the development and final usage. Consider for example the interaction between pgfplots and Ti*k*Z: one has always to be careful when defining specific options to the plot (e.g., the marks or number formatting) because messing up with the key paths may lead to compilation errors.

- The different types of diagrams can share keys when appropriate or not. Besides some keys that are very generic such as those pertaining to the color definition, some diagrams such as flow charts and circular diagram share the customization of the so called *module*, i.e., the single building block of the diagram. Conversely, diagrams like the bubble or the sequence diagram have unique keys for customization. Such a mix is a precise design choice: defining specific keys for each diagram would have been cleaner, but verbose and less intuitive for the final user.

- For the sake easy the code development, smartdiagram resorts to libraries similarly to Ti*k*Z. Specifically, the code is organized into definitions (containing the keys), styles (containing the styles that customize the aspect of the diagram having as input the keys) and the commands (containing the macro for building the diagram). While these libraries (code snippet below) are loaded by default, there is one that the user can use when needed, and it is explained in the next subsection.

```
\usesmartdiagramlibrary{%
core.definitions}
\usesmartdiagramlibrary{core.styles}
\usesmartdiagramlibrary{core.commands}
```

3.2 The additions library

The main purpose of the library is to allow the user to create annotations over a smart diagram, such as include additional arrows, modules or text. Indeed, the main limitation of the approach described in Section 2 is that a diagram built automatically from a list has little flexibility in terms of customization. Quite often, a user finds itself in the need of include further elements to the graphic. However, the modification of the simple mechanism of

```
\smartdiagram[<type>]{<list of items>}
```

is very difficult. How to add text or an additional graphical elements outside the diagram and in correspondence of specific blocks? The library `additions` takes precisely care of this problem.

The library provides two macros, one called `\smartdiagramadd` and a second one called `\smartdiagramconnect`. The first one replaces the basic `\smartdiagram` because it introduces an additional argument that defines the position of the new items with respect to the original diagram. Then, the second one, is employed to customize the connections between the additional elements and those of the basic diagram.

4 Some examples

This section overviews some examples of the operation of smartdiagram.

4.1 Traditional mode

First, this subsection overviews non-animated mode examples. This exposes how much it is simple to create such diagrams.

The first example is a basic usage:

```
% A new list of colors
\smartdiagramset{set color list={
blue!50!cyan,
green!60!lime,
orange!50!red},
bubble center node color=yellow!80!red
}
```

```
\begin{center}
\smartdiagram[bubble diagram]{
Cloud Computing, SaaS, PaaS, IaaS
}
\end{center}
```

that generates the output of Fig. 2.

The next example shows how to integrate additional elements. Specifically, these components are used to exemplify with a description the meaning of the titles used in the basic diagram. Notably, one of the strength of the library is that additional elements can be of an arbitrary number and this number does not have to match exactly with the number of items in the original diagram.

```
\usesmartdiagramlibrary{additions}%
% in the preamble

\smartdiagramset{set color list={
 blue!50!cyan,
 green!60!lime,
 orange!50!red}
}% A new list of colors

% An horizontal diagram:
% - first remove the arrow from
% the last block towards the first one
% - customize the aspect of the
% additional blocks
\smartdiagramset{
back arrow disabled=true,
additions={
additional item bottom color=%
orange!60!red!30,
additional item border color=gray,
additional item shadow=drop shadow,
additional item offset=0.65cm,
additional connections disabled=false,
additional arrow line width=2pt,
additional arrow tip=to,
additional arrow color=%
orange!60!red!50,
additional arrow style={]-latex},
 }
}

\begin{center}
\smartdiagramadd[%
flow diagram:horizontal]{
IaaS, PaaS, SaaS
}{
above of module1/
{Virtual machines, servers, storage,
load balancers, network},
above of module2/
{Execution runtime, database,
web server, development tools},
above of module3/
{CRM, Email, virtual desktop,
communication, games}
}
\end{center}
```

which leads to the result depicted in Fig. 3.

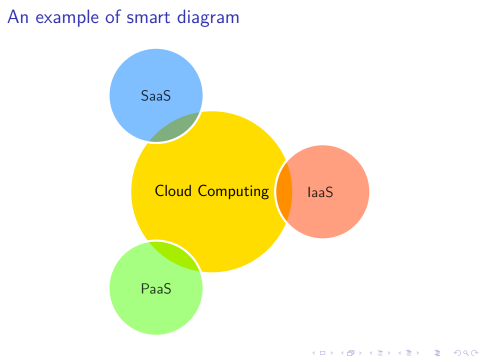


Figure 2: A example of bubble diagram

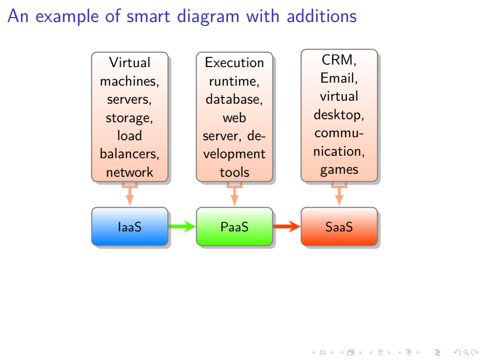

An example of smart diagram with additions
Virtual machines, servers, storage, load balancers, network
Execution runtime, database, web server, development tools
CRM, Email, virtual desktop, communication, games
IaaS
PaaS
SaaS

Figure 3: Including additional elements in the diagram

4.2 Animated mode

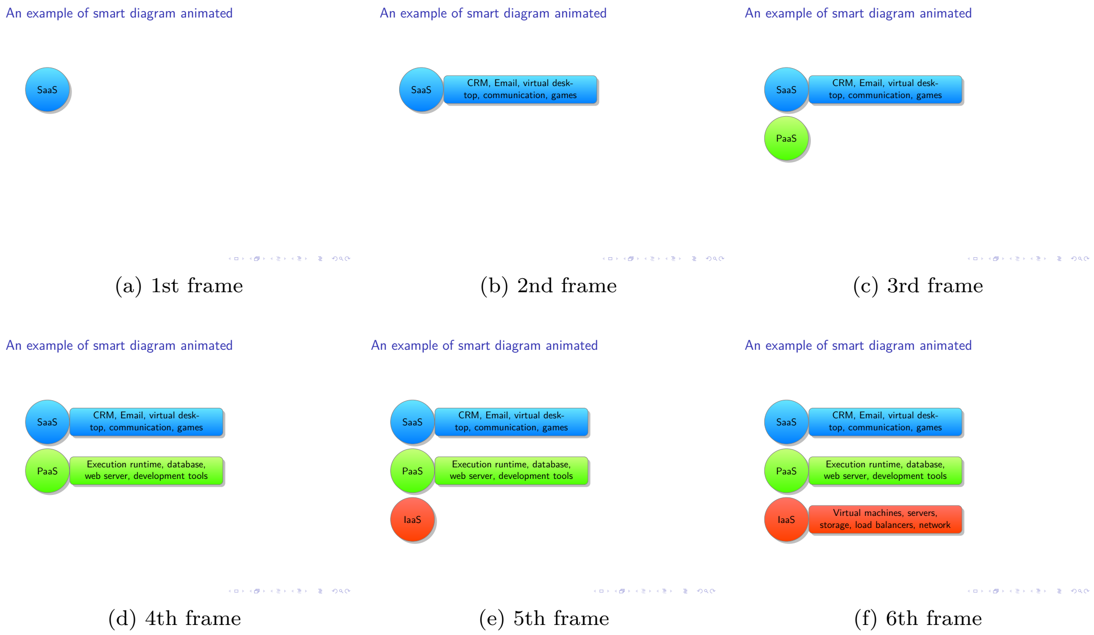
Another way to include descriptions is with the native `descriptive diagram`. This solution works well when the objective is to exemplify the meaning of each item in the original diagram.

```
\smartdiagramset{set color list={
 blue!50!cyan,
 green!60!lime,
 orange!50!red}
}% A new list of colors

% A descriptive diagram
\smartdiagramanimated[%
descriptive diagram]{
{SaaS,{CRM, Email, virtual desktop,
communication, games}},
{PaaS, {Execution runtime, database,
web server, development tools}},
{IaaS, {Virtual machines, servers,
storage, load balancers, network}},
}
```

The code produces the result depicted in Fig. 4 where each subfigure corresponds to the animation steps of each frame.

5 When it got out of control

I honestly admit to be extremely proud of the evolution of the package over the years. Smartdiagram has survived pretty well the *test of time* and it is quite used within the community. Specifically, it


An example of smart diagram animated
SaaS
CRM, Email, virtual desktop, communication, games
PaaS
Execution runtime, database, web server, development tools
IaaS
Virtual machines, servers, storage, load balancers, network

(a) 1st frame (b) 2nd frame (c) 3rd frame

(d) 4th frame (e) 5th frame (f) 6th frame

Figure 4: An example with animation

exists a tag for `smartdiagram`-related questions on TEX.stackexchange that contributed to its visibility, bug fixes and extensions.

Notably, use cases are listed in TEXample[2] and the LATEXCookbook[3]. It also exists a plugin for RMarkdown[4].

Smartdiagram was also employed in scientific publications in prestigious journals like IEEE Communication Surveys and Tutorials (mine - see Capponi *et al.* (2019), - and not - see Shit *et al.* (2019)).

Also one video tutorial on YouTube talks about the package and illustrates its operation[5].

6 Conclusion

This paper has presented the smartdiagram package and has exposed the motivation for its creation, the key idea behind its operation, the rationale behind the design choices and the journey of the package along the years. The paper has also highlighted some examples of diagrams that the package can build.

7 Acknowledgements

So many people to thank for this fantastic journey. Of course, first it goes the original poster on TEX.stackexchange: good works do not come for free, but in response of specific needs. Many thanks to the numerous contributors spotting bugs (sadly, I often run late in fixing them).

References

Capponi, A., C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich and P. Bouvry (2019). «A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities». *IEEE Communications Surveys Tutorials*, **21** (3), pp. 2419–2465.

Fiandrino, Claudio (2012). *Smartdiagram*. `http://www.ctan.org/pkg/smartdiagram`.

— (2014). «Realizzare semplici animazioni in figure: come usare TikZ in beamer». *ArsTEXnica*, (17), pp. 18–23. `http://www.guit.sssup.it/arstexnica/`.

Shit, R. C., S. Sharma, D. Puthal, P. James, B. Pradhan, A. van Moorsel, A. Y. Zomaya and R. Ranjan (2019). «Ubiquitous localization (ubiloc): A survey and taxonomy on device free localization for smart world». *IEEE Communications Surveys Tutorials*, pp. 1–33.

Tantau, Till (2010). *The TikZ and PGF Packages*. `http://www.ctan.org/pkg/pgf`.

▷ Claudio Fiandrino
IMDEA Networks Institute
`claudio dot fiandrino at imdea dot org`

[2] Available online at: `http://www.texample.net/tikz/examples/feature/smartdiagram/`

[3] Available online at: `http://latex-cookbook.net/articles/smart-diagrams/`

[4] Available online at: `https://edpflager.com/?p=4236`

[5] Available online at: `https://www.youtube.com/watch?v=1kY-1ssTk7o`

Metamorfosi dei tipi sublacensi

Claudio Vincoletto

Sommario

L'accurato recupero di un classico, nella sua prima versione digitale: un carattere tipografico fondato sulla calligrafia tardomedievale e impiegato nei primi incunaboli italiani, quindi riproposto agli inizi del XX secolo dall'ultima delle grandi *private presses* inglesi.

Abstract

The first and accurate digital revival of a classic typeface, based on the medieval calligraphy and used for the first time in italian incunables. A copy of this original was employed by the last representative of *private press movement.*

Fin da bambino provo un gusto innato per le cose strane e bizzarre, per la frequentazione delle zone liminari, varcandone poi spesso i confini allo scopo di inoltrarmi in quelle che gli antichi chiamavano *terrae monstruum.* Ricordo che, sfogliando i prediletti libri di storia naturale, rimanevo affascinato dal concetto di "anello di congiunzione", che animava i dibattiti fra i postulati della paleontologia e le teorie legate all'evoluzione. Come spiegare nello svolgersi del tempo alcuni ritrovamenti fossili che si presentavano dal nulla, come innovatori incontrastati di ciò che sarebbe venuto, tanto da mettere in crisi l'idea stessa di genealogia?

Tutto questo mi ha spinto ad affrontare uno dei caratteri più celebrati e, al tempo stesso, più problematici della storia della tipografia. Risale al 1465 il primo esemplare attestato di un libro stampato in Italia, a Subiaco, nei pressi di Roma. Giunti appena un anno prima, due monaci tedeschi — Conrad Sweynheym e Arnold Pannartz — per la prima volta sperimentarono quanto appreso nelle officine di Fust e Schöffer (allievi e soci di Gutenberg), adattando la forma delle lettere a tipi di scrittura più diffusi nella penisola e ispirati ai calligrafi umanisti, creando una sorta di ibrido oggi denominato Gotico-Antiqua (Boardley, 2019).

Diedero alle stampe solo quattro volumi con quel carattere, di cui solo tre sono pervenuti fino a noi (il *De oratore* di Cicerone, le *Opere* di Lattanzio, il *De civitate Dei* di Agostino). Trasferitisi a Roma nel 1467, e riallestendo totalmente la loro officina, ne forgiarono uno nuovo, comunemente ritenuto meno pregiato. Uno degli spunti primari del mio progetto venne fornito proprio da una copia del *De oratore* stampata a Subiaco, che è disponibile oggi in una pregevole edizione anastatica accompagnata da un'esauriente introduzione e una dettagliata scheda bibliografica (Cicero, 2015).

Per comprendere l'originalità di Sweynheym e Pannartz, va tenuto presente che quello finora discusso è un periodo che vede non solo lo sviluppo tecnologico delle modalità di scrittura (da quella manoscritta a quella stampata), ma che coincide anche con un'innovazione essenzialmente stilistica (il passaggio dalla forma gotica delle lettere a quella tonda) di cui il carattere forgiato a Subiaco rappresenta la transizione. Alcuni studi individuano, in diversi manoscritti presenti nei monasteri di Santa Scolastica e del Sacro Speco, le fonti che ispirarono i due prototipografi (ICCU, 2019). Si ritiene specificamente che per le lettere minuscole, più calligrafiche, il riferimento sia il ms. XXXIV, mentre le maiuscole sono riconducibili in modo più generico alle epigrafi in caratteri capitali e ai capilettera miniati presenti in alcuni manoscritti della biblioteca, come per esempio il ms. CLX (De Gregori, 1942).

Nel considerare la trasposizione dal modello calligrafico ai tipi in metallo, propongo sia lecito ritenere che il disegno dei glifi vero e proprio sia stato progettato in termini geometrici. In questa direzione, con l'ausilio di METAFONT, ho tentato di immaginare quali schemi potessero aver seguito i due monaci, convinto che la rappresentazione dello spazio bidimensionale della scrittura, così nel carattere tipografico come nella font digitale, non si potesse discostare di molto dalla trattatistica coeva dedicata alle forme dei caratteri, fortemente impregnata di modelli matematici. Tra i primi esempi risalta il lavoro di Felice Feliciano (*Alphabetum Romanum*), che precede di qualche anno le stampe di Subiaco. Tuttavia questi studi, così come quelli di Pacioli e Tory, si concentrano sulle lettere capitali romane, mentre personalmente ho trovato più utile un'opera di poco più tarda in cui vengono affrontati con questi criteri anche i caratteri minuscoli.

Il trattato di Sigismondo Fanti è infatti dedicato in buona parte alla conversione dei tratti calligrafici in coordinate geometriche (Fanti, 2013). Su questo modello, con pochissimi aggiustamenti, ho riscontrato delle ricorrenze nel carattere Subiaco che mi hanno convinto a fissare dei precisi rapporti a partire dalle dimensioni del corpo del carattere (in 6 mm), suddiviso in 21 unità (*hair*). Ho programmato dunque METAFONT in modo che vengano calcolate tutte le misure verticali di riferimento. Il medesimo principio viene poi adoperato anche

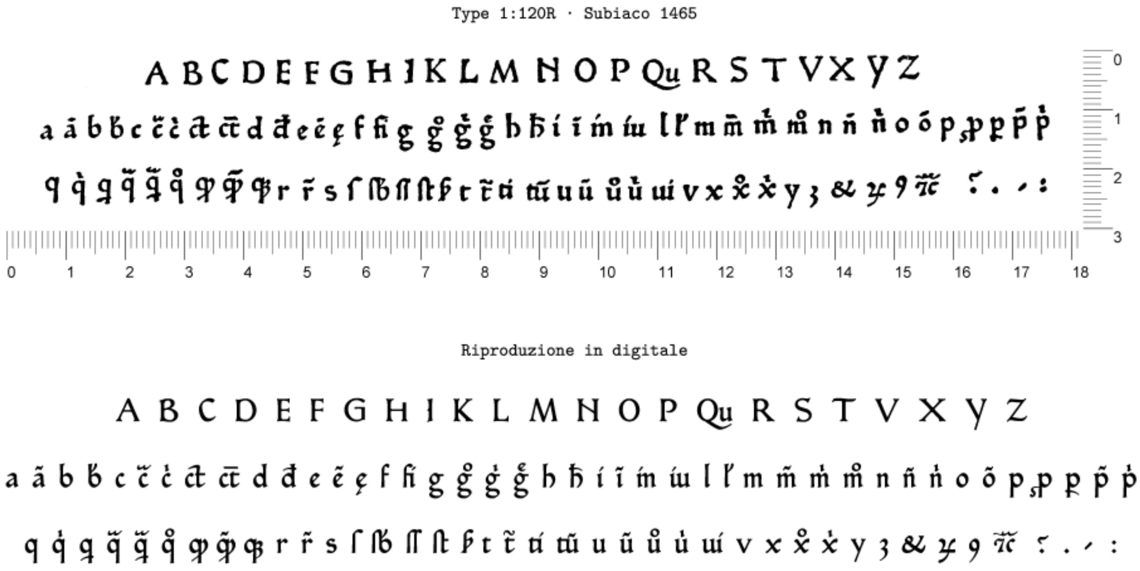


Figura 1: Confronto con i glifi originali (TW, 2019).

per la componente orizzontale, glifo per glifo:

```
mode_setup;

%size#:=6mm#;
size#:=17pt#;

font_size size#;
font_identifier:="Subiaco";
font_coding_scheme:="TeX text";

% Proportions
hair# = 1/21 size#;
body_height# + body_depth# = size#;
body_height# = 14/21 size#;
x_height# = 8/21 size#;
asc_height# = 13.5/21 size#;
cap_height# = 13/21 size#;
bar_height# = 6.5/21 size#;
desc_depth# = 6/21 size#;
```

Non mi è possibile dimostrare che Sweynheym e Pannartz abbiano impiegato deliberatamente la successione di Fibonacci nelle proporzioni del disegno. Tuttavia, la suggestione è molto forte e può essere ricondotta alla diffusa mentalità condivisa dai sapienti dell'epoca. I recenti studi di Frank Blokland vanno in questa direzione, come ho avuto modo di accennare diversi anni fa, sebbene la sua ricerca non fosse conclusa (Vincoletto, 2013). Egli ipotizza che agli inizi della stampa vi sia stata l'esigenza di una normalizzazione dei tipi gotici, poi convertita anche nella fabbricazione dei tipi romani. Questo sarebbe evidente nelle relazioni morfologiche che sottostanno alla struttura di tali alfabeti. Pertanto, le convenzioni tipografiche si sarebbero sviluppate a partire da schemi ricorrenti, costruiti su modelli armonico-ritmici, completamente estranei a preferenze ottiche. A suo avviso, sarebbe in questo modo possibile ottenere maggiori informazioni sul processo creativo dei caratteri antichi, se non addirittura individuare modelli parametrici per la progettazione di caratteri digitali (Blokland, 2016).

L'utilizzo di METAFONT si rivela molto efficace nel costruire i glifi seguendo queste indicazioni, proprio in virtù del suo carattere essenzialmente parametrico. Tuttavia, per rendere alcuni aspetti della componente calligrafica del Subiaco ho dovuto modificare alcuni comportamenti del programma, forzandolo in due direzioni. Una delle mie priorità consisteva nell'elaborare il disegno usando esclusivamente dei contorni; l'altra, nel recuperare in modo obliquo l'idea delle penne virtuali proposta da Knuth. Il metodo adoperato si basa sul concetto di "interpolazione", per cui una volta fissata una serie di punti estremi, che sintetizzano l'andamento di un tratto tracciato da una penna, questi vengono collegati fra loro in una linea ciclica mediante delle curve di Beziér. Per rendere evidente l'applicazione di questi passaggi, viene riportato per intero il codice che una volta compilato genera la lettera "b", illustrata nella figura 2.

```
"Subiaco letter b";
beginsubiacochar("b",9,asc_height#,0);
penpos1(curve,pa); penpos2(curve,pa);
penpos3(curve,pa); penpos4(.9curve,pa);
penpos41(curve,pa); penpos12(curve,pa);
penpos23(curve,pa); penpos34(curve,pa);
penpos5(stem,pa); penpos6(stem,pa);
x1=x3=.5[x4,x2]; x2=w-1.25hair;
x4=x5=2hair; x6=1.9hair;
y1m=x_height+o; y3l=y6l=0-o;
y2=y4=.5[y3,y1]; y5=h;
p1=z1{right}..z2{down}..z3{left}..z4{up}..cycle;
z41= directionpoint dir pa of p1;
z12= directionpoint dir (pa-90) of p1;
z23= directionpoint dir (pa-180) of p1;
z34= directionpoint dir (pa-270) of p1;
curvestroke12(1,12,2,blob);
curvestroke23(2,23,3,1.5blob);
curvestroke34(3,34,4,blob);
z4'=whatever[z5,z6]; y4'=.6875x_height;
```

```
numeric theta;
theta:=angle((x1,y1+.125x_height)-z4');
penpos4'(thin,theta-90);
penstroke56(5,6,.25,.48,.75,flx);
dish_serif(6,5,a,.7slab,jut,1.6jut);
p2=(dserif5 rotatedaround(z5,pa_f));
p3=z4'l{dir theta}..tension 3 and 1..{right}z1m;
p4=z1l{left}..tension 1 and 3..{-dir theta}z4'r;
p5=stroke56.r...z6r;
numeric i,j,g,f,r,s,t,u;
(t,u) = p3 intersectiontimes p5;
(s,r) = p4 intersectiontimes p5;
(i,j) = arc34.r intersectiontimes p5;
fill stroke56l...p2...subpath(0,u-.3blob) of p5..
  subpath(t+.3blob,length p3) of p3 & arc12.r &
  arc23.r & subpath(0,1.5) of arc34.r...cycle;
unfill arc34.l & arc23.l & arc12.l &
subpath(0,s-.5blob) of p4..cycle;
penlabels(range 1 thru 44,4');
endchar;
```

Si è detto quanto il Subiaco porti in sé la commistione di elementi tipografici (regolarità e uniformità del tratto) ed elementi calligrafici (inclinazioni, *ductus* della penna). Pertanto, in una prima fase il lavoro si è concentrato sull'individuazione, mediante sintesi, dei componenti costitutivi di ogni tratto, da cui attingere per assemblare lettera per lettera. Nella costruzione del glifo in figura, infatti, si possono identificare almeno due tratti essenziali: uno verticale, l'asta a sinistra, e l'altro curvo, che costituisce l'arco, altrimenti detto "pancia", della lettera. In genere, per simulare un tratto calligrafico con METAFONT, si usa l'istruzione `draw`, ma in questo caso sono stati ridefiniti alcuni automatismi di base del programma, creando apposite definizioni (*macro*). Sia d'esempio la definizione `penstroke`, per determinare i contorni delle linee diritte, di cui si darà una breve descrizione:

```
vardef penstroke@#(suffix $,$$)
(expr start_point,mid_point,end_point,curve) =
  z@#h.r=point start_point of (z$r--z$$r);
  z@#m.r=point mid_point of (z$r--z$$r);
  z@#l.r=point end_point of (z$r--z$$r);
  z@#h.n=point start_point of (z$n--z$$n);
  z@#m.n=point mid_point of (z$n--z$$n);
  z@#l.n=point end_point of (z$n--z$$n);
  if y$=y$$:
    if y$l>y$r:
    y@#m.r:=y@#m.r+curve;
    y@#m.n:=y@#m.n-curve;
    else:
    y@#m.r:=y@#m.r-curve;
    y@#m.n:=y@#m.n+curve;
    fi
  else:
    if x$l<x$r:
    x@#m.r:=x@#m.r-curve;
    x@#m.n:=x@#m.n+curve;
    else:
    x@#m.r:=x@#m.r+curve;
    x@#m.n:=x@#m.n-curve;
    fi
  fi
  stroke[@#].r:=z@#h.r..z@#m.r{z$$r-z$r}..z@#l.r;
  stroke[@#].l:=z@#l.n..z@#m.n{z$n-z$$n}..z@#h.n;
  labels(@#h,@#m,@#l,@#m.r,@#m.n,@#h.r,
        @#h.n,@#l.r,@#l.n);
enddef;
path stroke[].l, stroke[].r;
```

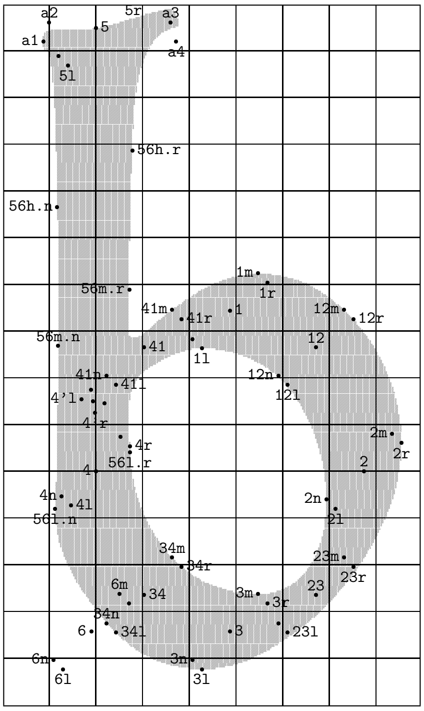


Figura 2: Particolare della lettera *b*.

Per valutare l'inserimento dei parametri nella definizione `penstroke`, si vedano le prime due linee del codice riportato. Ogni tratto rettilineo va distinto per essere riconosciuto dal sistema ed è pertanto numerato con (`@#`). Seguono i punti estremi della linea, quello iniziale (`$`) e quello finale (`$$`). Quindi l'indicazione, in decimale rispetto alla lunghezza della linea, dell'effettivo punto di partenza del contorno (`start_point`), quello intermedio (`mid_point`) e quello di arrivo (`end_point`). Le ragioni che portano a individuare un punto intermedio sono dovute alle caratteristiche del tratto calligrafico, per cui questo risulta tendenzialmente più spesso alle estremità. Donald Knuth ne illustra un caso generico a p. 28 nel suo *The METAFONTbook* (Knuth, 1986). Tale punto di flessione risulta anche congeniale in fase di stampa, in quanto rende l'asta meno bombata e maggiormente uniforme. Il divario dello spessore del punto intermedio rispetto alle estremità è indicato dal valore espresso da `curve`.

Nel caso riportato, in merito alla lettera *b*, le istruzioni impartite sono le seguenti (`flx` corrisponde a 2/1000 di `size`):

```
penstroke56(5,6,.25,.48,.75,flx);
```

Un approccio differente si può incontrare nella definizione `curvestroke`, in cui il contorno del tratto curvo è ricostruito per quadranti. Oltre ai punti ortogonali, viene automaticamente assegnato un punto intermedio sull'arco definito, in funzione dell'angolatura della penna. Anche in questo caso propongo l'analisi del codice impiegato:

```
vardef curvestroke@#(suffix $,$$,$$$)(expr blob) =
numeric sect[].t, sect[].u;
  if y$>y$$$:
    if x$<x$$$:
% quadr 1
    arc[@#].r:=subpath(0,1-blob) of (z$.m{right}..
      z$$.m)..subpath(blob,1) of (z$$.r..
      z$$$.r{down});
      (sect@#.t,sect@#.u) = (z$$$.n{up}..z$$.n)
      intersectiontimes (z$$.l..z$.l{left});
      arc[@#].l:=subpath(0,sect@#.t-blob) of
      (z$$$.n{up}..z$$.n)...
      subpath(sect@#.u+blob,1) of (z$$.l..
      z$.l{left});
    else:
% quadr 4
    arc[@#].r:=subpath(0,1-blob) of (z$.r{down}..
      z$$.r)...subpath(blob,1) of (z$$.l..
      z$$$.l{left});
      (sect@#.t,sect@#.u) = (z$$$.m{right}..z$$.m)
      intersectiontimes (z$$.n..z$.n{up});
    arc[@#].l:=subpath(0,sect@#.t-blob) of
      (z$$$.m{right}..z$$.m)...
      subpath(sect@#.u+blob,1) of (z$$.n..z$.n{up});
    fi
  else:
    if x$>x$$$:
% quadr 3
    arc[@#].r:=subpath(0,1-blob) of (z$.l{left}..
      z$$.l)...subpath(blob,1) of (z$$.n..
      z$$$.n{up});
      (sect@#.t,sect@#.u) = (z$$$.r{down}..z$$.r)
      intersectiontimes (z$$.m..z$.m{right});
    arc[@#].l:=subpath(0,sect@#.t-blob) of
      (z$$$.r{down}..z$$.r)...
      subpath(sect@#.u+blob,1) of (z$$.m..
      z$.m{right});
    else:
% quadr 2
    arc[@#].r:=subpath(0,1-blob) of (z$.n{up}..
      z$$.n)...subpath(blob,1) of (z$$.m..
      z$$$.m{right});
      (sect@#.t,sect@#.u) = (z$$$.l{left}..z$$.l)
      intersectiontimes (z$$.r..z$.r{down});
      arc[@#].l:=subpath(0,sect@#.t-blob) of
      (z$$$.l{left}..
      z$$.l)...subpath(sect@#.u+blob,1) of
      (z$$.r..z$.r{down});
    fi
  fi
enddef;
path arc[].l, arc[].r;
```

Si può notare come, oltre all'identificatore (`@#`), vengano indicati il punto di partenza (`$`), quello mediano (`$$`) e quello terminale (`$$$`). Si aggiunge inoltre un valore al parametro `blob`, che esprime la distanza sottratta rispettivamente ai due percorsi secanti, a partire dal loro punto di intersezione, per rendere una curva più morbida e senza spigoli. In tal modo, la soluzione assegnata al primo quadrante della lettera "b" è il seguente:

```
curvestroke12(1,12,2,blob);
```

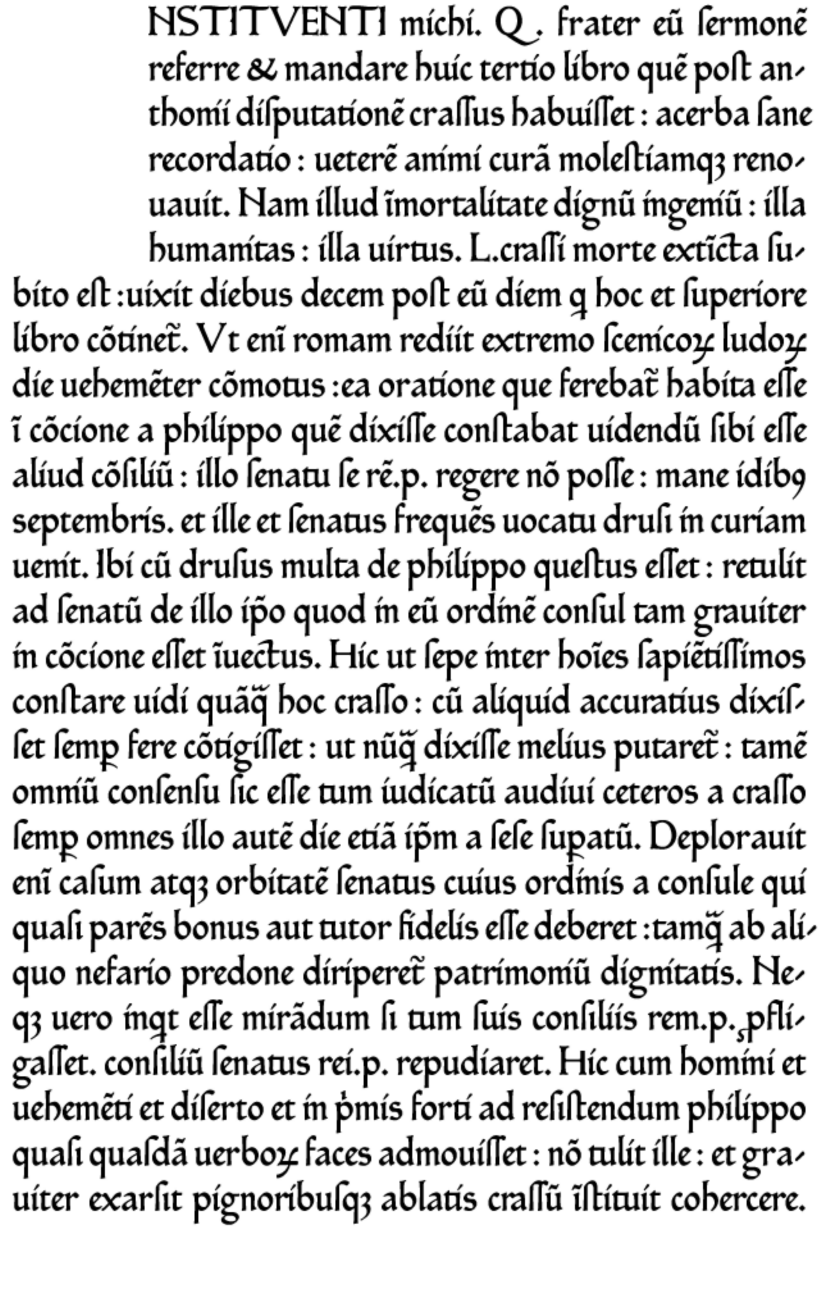
Potrebbero essere fornite ancora diverse indicazioni relative al codice della lettera "b", ma è necessaria una digressione storica per discutere le ragioni di alcuni accorgimenti adottati.

Il recupero di caratteri del passato non è una concezione moderna. Nel periodo umanistico, e successivamente con l'introduzione dei tipi romani, i riferimenti erano la scrittura carolina per il minuscolo e la capitale quadrata per il maiuscolo. Come già affermato, la grafia impiegata nei manoscritti venne studiata secondo una prospettiva geometrica, e questo in qualche modo contribuì a semplificare la conversione della scrittura in funzione dell'innovazione tecnologica portata dall'invenzione della stampa a caratteri mobili.

Alla fine del XIX secolo, il movimento Arts and Crafts, promosso da William Morris, segna una nuova fase di questo processo, con un pionieristico approccio al disegno dei caratteri mediante la fotografia. Stimolato dal gusto critico di John Ruskin, egli condivise il proposito di ricorrere al Medioevo come fonte di ispirazione, non tanto per uno spirito reazionario da contrapporre alla modernità portata dall'industrializzazione, quanto per arginare un diffuso decadimento dei valori estetici con il recupero e la riattualizzazione storica di un patrimonio perduto. Tra le arti applicate, la tipografia assunse un ruolo peculiare, e la Kelmscott Press fondata da Morris divenne un modello di perfezione formale, che venne imitato dalle piccole stamperie private inglesi nate a cavallo fra i due secoli. Oltre all'elevata qualità delle carte, degli inchiostri e dei torchi, Morris attinse dagli incunaboli la progettualità grafica, l'impostazione della pagina, le decorazioni e il disegno di alcuni tipi che fece riprodurre per le sue edizioni. Inizialmente fu sua l'idea di recuperare nel 1892 il carattere forgiato a Subiaco e riproporlo per la propria casa editrice, ma il progetto non vide mai la luce (Cockerell, 1898).

Solamente nel 1902 venne realizzata una vera e propria riproduzione in metallo del carattere Subiaco. L'opera fu commissionata da Charles St John Hornby per la sua Ashendene Press, e messa a punto da un gruppo di persone che avevano strettamente lavorato con Morris: Emery Walker fornì degli ingrandimenti fotografici tratti dalle opere di Sweynheym e Pannartz, mentre Sidney Cockerell ne abbozzò il disegno. L'incisione venne portata a termine da Edward Prince, che già aveva intagliato tutti i caratteri utilizzati prima dalla Kelmscott Press e poi dalla Doves Press. Per le iniziali calligrafiche delle sue pubblicazioni, la Ashendene si avvalse dei più celebri artisti dell'epoca: Graily Hewitt, Edward Johnston ed Eric Gill. Questo per rimarcare la linea di continuità che legò i progetti editoriali di Kelmscott, Doves e Ashendene Press.

Il disegno del Subiaco di St John Hornby si presenta molto fedele se affiancato a quello del 1465, e questo dipende probabilmente dai principi usati per adattarlo (McNeil, 2017). In questo senso, è interessante prendere in considerazione alcuni saggi di Hermann Zapf, in cui viene affrontato il tema del revival, proprio discutendo le metodiche fotografiche adottate in quel periodo. In *Caratteri da stampa e libri* (1962) e nelle *Trasformazioni nel di-*



NSTITVENTI michi. Q. frater eũ ſermonẽ referre & mandare huic tertio libro quẽ poſt anthonii diſputationẽ craſſus habuiſſet: acerba ſane recordatio: ueterẽ animi curã moleſtiamqꝫ renouauit. Nam illud ĩmortalitate dignũ ingeniũ: illa humanitas: illa uirtus. L.craſſi morte extĩcta ſubito eſt: uixit diebus decem poſt eũ diem ꝗ hoc et ſuperiore libro cõtinet̃. Vt enĩ romam rediit extremo ſcenicoꝝ ludoꝝ die uehemẽter cõmotus: ea oratione que ferebat̃ habita eſſe ĩ cõcione a philippo quẽ dixiſſe conſtabat uidendũ ſibi eſſe aliud cõſiliũ: illo ſenatu ſe rẽ.p. regere nõ poſſe: mane idibꝰ ſeptembris. et ille et ſenatus frequẽs uocatu druſi in curiam uenit. Ibi cũ druſus multa de philippo queſtus eſſet: retulit ad ſenatũ de illo ip̃o quod in eũ ordinẽ conſul tam grauiter in cõcione eſſet ĩuectus. Hic ut ſepe inter hoĩes ſapiẽtiſſimos conſtare uidi quãq̃ hoc craſſo: cũ aliquid accuratius dixiſſet ſemꝑ fere cõtigiſſet: ut nũq̃ dixiſſe melius putaret̃: tamẽ omniũ conſenſu ſic eſſe tum iudicatũ audiui ceteros a craſſo ſemꝑ omnes illo autẽ die etiã ip̃m a ſeſe ſuꝑatũ. Deplorauit enĩ caſum atqꝫ orbitatẽ ſenatus cuius ordinis a conſule qui quaſi parẽs bonus aut tutor fidelis eſſe deberet: tamq̃ ab aliquo nefario predone diriperet̃ patrimoniũ dignitatis. Neqꝫ uero inꝗt eſſe mirãdum ſi tum ſuis conſiliis rem.p. ꝓfligaſſet. conſiliũ ſenatus rei.p. repudiaret. Hic cum homini et uehemẽti et diſerto et in ṗmis forti ad reſiſtendum philippo quaſi quaſdã uerboꝝ faces admouiſſet: nõ tulit ille: et grauiter exarſit pignoribuſqꝫ ablatis craſſũ ĩſtituit cohercere.

Figura 3: Simulazione di una pagina del *De Oratore*, Subiaco 1465 (BVMC, 2019).

segno dei caratteri in seguito all'evoluzione tecnica (1967), Zapf ripercorre la storia del libro stampato, da Gutenberg all'inizio del XX secolo, equiparando tre diversi caratteri, tutti ispirati alla medesima fonte, ossia i tipi di Nicolas Jenson (Zapf, 1991). Il Golden Type della Kelmscott, risalente al 1890, rappresenta una pietra miliare della storia della tipografia, per i motivi spiegati innanzi, ma secondo Zapf è pure un fallimento. Gli ingrandimenti fotografici erano difettosi, forse per una carta bagnata eccessivamente, per cui il ricalco del contorno dei glifi risulta molto diverso dall'originale quattrocentesco. Morris ne era consapevole, tuttavia era interessato più all'effetto unitario dato dal colore scuro, quasi gotico, delle sue stampe, piuttosto che a un vero recupero filologico. Il Doves Type fu sviluppato tra il 1899 e il 1901 da Emery Walker e il suo assistente Percy Tiffin, su commissione di Thomas Cobden-Sanderson. Nuove e più accurate impressioni fotografiche consentirono di disegnare un carattere più leggero e maggiormente vicino a quello di Jenson, anche se con un profilo nettamente più moderno. Infine il Centaur, dapprima inciso privatamente dalla American Type Founders e quindi commercializzato dalla Monotype, risale a qualche anno dopo. Anche in questo caso Bruce Rogers fece ricorso alle fotografie, ma tracciandovi sulla superficie dei tratti con penna piatta diede al suo carattere un'impronta molto più calligrafica. Le diverse tecniche analizzate da Zapf fanno intendere la problematicità delle scelte progettuali messe in campo all'epoca, così come del resto accade ancora oggi.

Per il lavoro al Subiaco digitale, oltre a documenti disponibili in rete, mi sono avvalso delle fotografie di alta qualità fornitemi gentilmente da Riccardo Olocco, che ritraggono frammenti delle *Opere* di Lattanzio del 1465, conservate presso la Biblioteca Civica di Verona. Sebbene le tecniche digitali abbiano cambiato molto il modo di lavorare su supporti fotografici, l'adattamento necessario alla creazione di un carattere richiede approcci non dissimili da quelli del passato. Riprodurre esattamente quanto fatto secoli or sono è un'impresa disperata, sempre che quello sia il vero obiettivo. Un revival digitale non solo deve tener conto di aspetti estetici, ma anche essere rapportato alle tecnologie di uso corrente, per un impiego che sia al tempo stesso efficace e pertinente. Questi argomenti vengono affrontati da Paul Shaw nella discussione circa il trasporto in digitale di caratteri del passato (Shaw, 2017). Nel percorso storico che viene tracciato, sono presentati sia il Golden Type, digitalizzato dalla ITC, che il Doves Type, un faticoso recupero promosso da Robert Green e ultimato nel 2015. Un approfondimento sulle operazioni messe in campo per la riproduzione del Golden Type vengono discusse da Helga Jörgensen e Sigrid Engelmann, e alcune soluzioni intraprese sono tenute presenti nelle operazioni di costruzione del Subiaco (Karow, 1994). Ad esempio, la modalità in cui vengono gestite le intersezioni fra le diverse linee del contorno (*path*). Ritornando al codice della lettera "b", queste vengono risolte individuando il momento di incontro fra due percorsi piuttosto che le relative coordinate. Con METAFONT è possibile specificare tale punto, tenendo presente questo principio:

```
numeric t,u;
(t,u) = path_a intersectiontimes path_b;
```

Successivamente, sarà quindi possibile sottrarre ai rispettivi percorsi una piccola frazione di ognuno, con il parametro `blob`, e collegare così le porzioni rimanenti con una curva, eliminando di fatto lo spigolo dell'intersezione. In questo modo si può ottenere una transizione morbida da un tracciato all'altro, simulando da un lato le macchie d'inchiostro nel processo di stampa a torchio (*ink spread*) e favorendo dall'altro la conversione dell'immagine da vettoriale a bitmap per la stampa digitale:

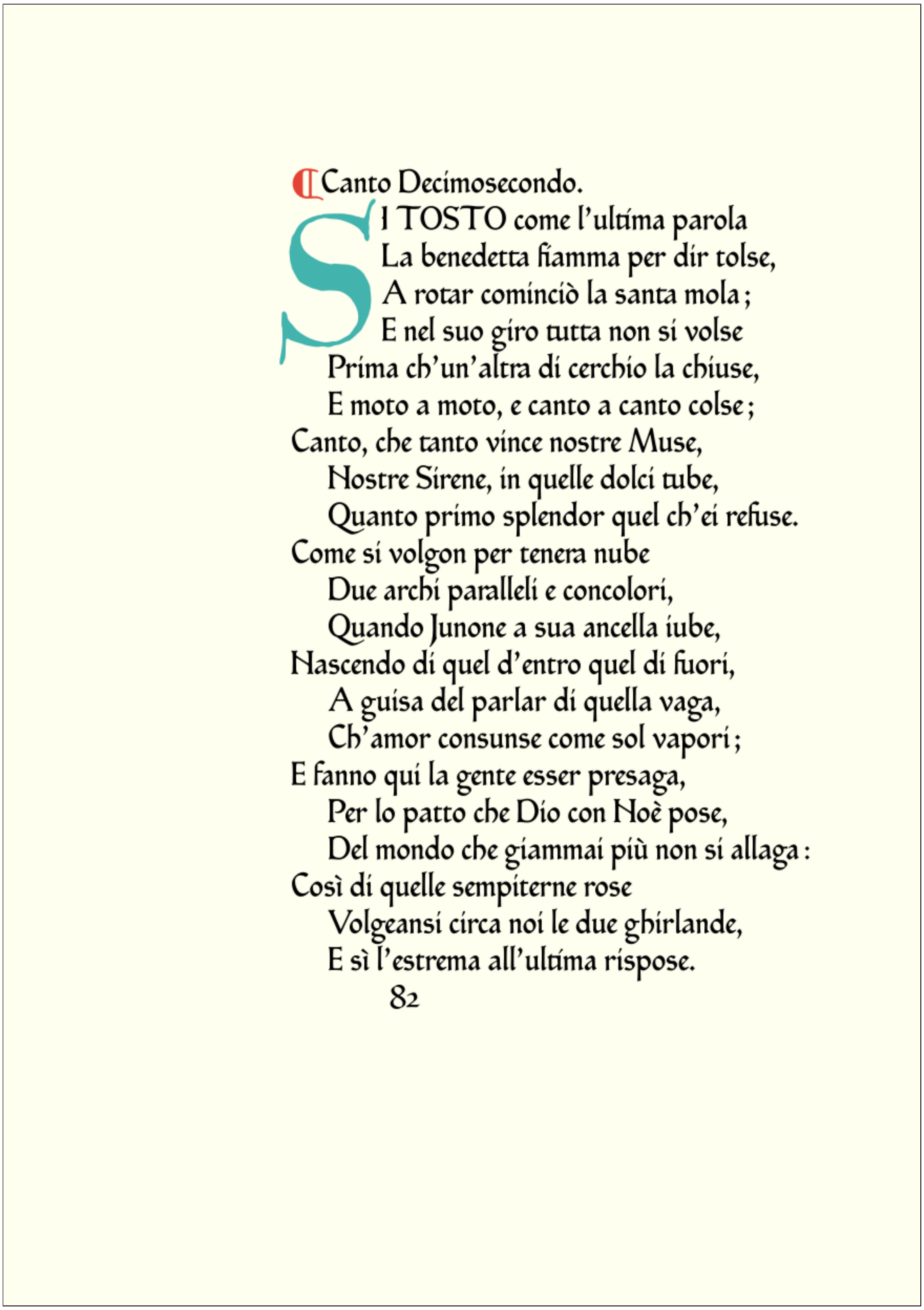
```
  subpath(0,t-blob) of path_a ...
  subpath(u+blob,infinity) of path_b;
```

Riprendendo il discorso sul Subiaco della Ashendene Press, bisogna considerare che il carattere fu adattato per comporre testi in lingue moderne. Mi è sembrato adeguato perciò estendere l'alfabeto del Subiaco digitale, seguendo gli stessi criteri, anzi ampliandone le possibilità con diacritici e punteggiature diffuse nelle varie lingue europee. Per semplificare l'operazione, anche in questo caso sono ricorso a definizioni in grado di riprodurre la stessa figura in diversi glifi. Per esempio, nel caso dell'*accento acuto*, distribuito sulle vocali minuscole e maiuscole, si pone l'esigenza di collocare il diacritico in una posizione precisa, in base a un punto di riferimento riconducibile alla lettera stessa. Ne riporto il codice esplicativo:

```
vardef acute_acc(suffix $)(expr xacc,yacc) =
x$=xacc; x$'=x$+2.75hair;
y$=yacc-.125x_height; y$'=yacc+.125x_height;
numeric delta; delta:=angle(z$-z$');
penpos$(1.7thin,delta-90);
penpos$'(3.7thin,delta-90);
acute$:=z$'l---z$n..tension 2..z$m---z$'r
       ..tension 2..cycle;
penlabels($,$');
enddef;
path acute[];
```

Con il suffisso `$` viene creato un nuovo punto nello spazio, di cui a seguire vengono indicate le coordinate, `xacc` e `yacc`, in relazione ai parametri riferiti al glifo su cui poggia. Quindi viene generato un percorso ciclico tra due punti a esso correlati. L'istruzione per generare un accento acuto sommato a una vocale è pertanto la seguente:

```
acute_acc(15,2.25hair,acc_height);
fill acute15;
```



¶ Canto Decimosecondo.
SI TOSTO come l’ultima parola
La benedetta fiamma per dir tolse,
A rotar cominciò la santa mola;
E nel suo giro tutta non si volse
Prima ch’un’altra di cerchio la chiuse,
E moto a moto, e canto a canto colse;
Canto, che tanto vince nostre Muse,
Nostre Sirene, in quelle dolci tube,
Quanto primo splendor quel ch’ei refuse.
Come si volgon per tenera nube
Due archi paralleli e concolori,
Quando Junone a sua ancella iube,
Nascendo di quel d’entro quel di fuori,
A guisa del parlar di quella vaga,
Ch’amor consunse come sol vapori;
E fanno qui la gente esser presaga,
Per lo patto che Dio con Noè pose,
Del mondo che giammai più non si allaga:
Così di quelle sempiterne rose
Volgeansi circa noi le due ghirlande,
E sì l’estrema all’ultima rispose.
82

Figura 4: Una pagina della *Commedia* di Dante, nello stile della Ashendene Press (Knight, 2012).

Per generare un file OpenType dai sorgenti METAFONT sono ricorso a `mf2outline` di Linus Romer, uno script Python in grado di invocare dapprima METAPOST come sostituto di METAFONT per la sintesi dei contorni, quindi FontForge di George Williams per la compilazione del file `.otf`. Successivamente, sempre con FontForge, ho aggiunto un *feature file*, che rende possibile effettuare automaticamente la sostituzione dei glifi (ad esempio le legature) oltre a sporadiche correzioni di spaziatura fra caratteri. La struttura rigorosa della componente orizzontale dei glifi, come specificato in precedenza, basata sulla ripetizione ritmica di unità (*cadence units*, nel lessico di Blokland) è in grado di riprodurre con fedeltà la successione dei caratteri sia del Subiaco originale che della trasposizione della Ashendene. Le correzioni di crenatura, ridotte a poche coppie di glifi, sono riconducibili ai medesimi parametri.

Il risultato finale è una font digitale che, malgrado le numerose varianti, non dispone di tutti i glifi di un alfabeto moderno. Non è corredata di corsivo, grassetto, maiuscoletto e via dicendo. Consente, però, di compilare dei testi in stile austero e comunque sufficientemente leggibili da risultare gradevoli. Come ho illustrato in questo scritto, il Subiaco è un tipo che già all'origine manifesta componenti ibride, che vengono mantenute in tutte le sue trasformazioni. Gli stessi approcci, nonché gli strumenti da me messi in opera per realizzarlo, sono eterogenei e multiformi, in una sinergia che può svelare il suo valore solo nei risultati ottenuti. Il fatto stesso che riviva oggi in questa trasposizione digitale ci fa capire quanto il Subiaco sia un classico, e come il modello archetipico travalichi il tempo e le situazioni.

Riferimenti bibliografici

Blokland, F.E. (2016). *On the origin of patterning in movable Latin type: Renaissance standardisation, systematisation, and unitisation of textura and roman type.* Tesi di Dottorato, Leiden University. `https://openaccess.leidenuniv.nl/handle/1887/43556`.

Boardley, J. (2019). *Typographic Firsts. Adventures in Early Printing.* Bodleian Library.

BVMC (2019). «Cicero, De Oratore, p.159. Biblioteca Virtual Miguel de Cervantes». `http://www.cervantesvirtual.com/obra-visor/de-oratore--0/html/01bdc2d6-82b2-11df-acc7-002185ce6064_166.html`.

Cicero, M.T. (2015). *De Oratore. Subiaco 1465. Ristampa anastatica.* Iter Edizioni.

Cockerell, S.C. (1898). *A note by William Morris on his aims in founding the Kelmscott Press.* Kelmscott Press. `https://archive.org/details/ANoteByWilliamMorrisOnHisAimsInFoundingTheKelmscottPressTogether`.

De Gregori, L. (1942). «I tipi sublacensi». *Studi e ricerche sulla storia della stampa del Quattrocento*, pp. 47–61.

Fanti, S. (2013). *Trattato di scrittura. Theorica et pratica de modo scribendi (Venezia 1514).* Salerno Editrice.

ICCU (2019). «ManusOnLine. Istituto Centrale per il Catalogo Unico.» `https://manus.iccu.sbn.it/opac_ElencoSchedeDiUnFondo.php?ID=145`.

Karow, P. (1994). *Font Technology. Descriptions and Tools.* Springer-Verlag.

Knight, S. (2012). *Historical Types. From Gutenberg to Ashendene.* Oak Knoll Press.

Knuth, D.E. (1986). *The METAFONTbook.* Computers & typesetting. Addison-Wesley.

McNeil, P. (2017). *The Visual History of Type.* Laurence King Publishing.

Shaw, P. (2017). *Revival Type. Digital typefaces inspired by the past.* Yale University Press.

TW (2019). «Type 1:120R. Typenrepertorium der Wiegendrucke. Staatsbibliothek zu Berlin». `https://tw.staatsbibliothek-berlin.de/ma10000`.

Vincoletto, C. (2013). «Gli alfabeti romani di Francesco Griffo». *ArsTEXnica*, (16), pp. 52–57. `http://www.guitex.org/home/numero-16`.

Zapf, H. (1991). *Dalla calligrafia alla fotocomposizione.* Edizioni Valdonega.

▷ Claudio Vincoletto
Torino
claudio dot vincoletto at
gmail dot com

Questa rivista è stata prodotta
dal Gruppo Utilizzatori Italiani di TEX
usando esclusivamente software libero.

Versione elettronica per la diffusione via web.

ArsTeXnica – Call for Paper

La rivista è aperta al contributo di tutti coloro che vogliano partecipare con un proprio articolo. Questo dovrà essere inviato alla redazione di ArsTeXnica, per essere sottoposto alla valutazione di recensori entro e non oltre il 14 Febbraio 2020. È necessario che gli autori utilizzino la classe di documento ufficiale della rivista; l'autore troverà raccomandazioni e istruzioni più dettagliate all'interno del file d'esempio (.tex).
Gli articoli potranno trattare di qualsiasi argomento inerente al mondo di LaTeX e non dovranno necessariamente essere indirizzati ad un pubblico esperto. In particolare tutorial, rassegne e analisi comparate di pacchetti di uso comune, studi di applicazioni reali, saranno bene accetti, così come articoli riguardanti l'interazione con altre tecnologie correlate.
Di volta in volta verrà fissato, e reso pubblico sulla pagina web `http://www.guitex.org/arstexnica/`, un termine di scadenza per la presentazione degli articoli da pubblicare nel numero in preparazione della rivista. Tuttavia gli articoli potranno essere inviati in qualsiasi momento e troveranno collocazione, eventualmente, nei numeri seguenti.
Chiunque, poi, volesse collaborare con la rivista a qualsiasi titolo (recensore, revisore di bozze, grafico, etc.) può contattare la redazione all'indirizzo arstexnica@guitex.org.

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 28, Ottobre 2019

90028

9 771828 236001