Numero 16
Ottobre
2013

ArsTEXnica

Rivista italiana di TEX e LATEX

Atti del GUIT *meeting 2013*

GUIT

http://www.guitex.org/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@sssup.it`.

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 Ufficio Statistica
Scuola Superiore Sant'Anna
Piazza Martiri della Libertà 33
56127 Pisa, Italia.
`http://www.guitex.org`
`guit@sssup.it`

Redazione ArsTEXnica:
`http://www.guitex.org/arstexnica/`
`arstexnica@sssup.it`

Codice ISSN 1828-2369

Stampata in Italia
Pisa: 15 Ottobre 2013

In quest caso, aver udit né tra' m storici, n esecrabile lamente l le fu cos pur mi ve pensarlo, con qual io o poss re la turp

GUIT*meeting* 2013

DECIMO CONVEGNO NAZIONALE SU TEX, LATEX E TIPOGRAFIA DIGITALE

26 ottobre 2013

Roma, Facoltà di Ingegneria, *via Eudossiana, 18*
Sala del chiostro
Sapienza Università di Roma

Programma del convegno

10:00 Benvenuto. Inizio dei lavori.

10:10 *Scrivere la tesi di laurea in LaTeX*. Agostino De Marco.
10:50 *LaTeX e le lingue romanze alpine: il piemontese*. Claudio Beccari.

11:30 — Pausa caffè (30').

12:00 *LaTeX e la documentazione di progetto*. Roberto Giacomelli.
12:40 *Gli alfabeti romani di Francesco Griffo*. Claudio Vincoletto.

13:20 — Pausa pranzo (90').

14:50 *Il pacchetto* minerva. Grazia Messineo, Salvatore Vassallo.
15:30 *Experiments with multitasking and multithreading in LuaTeX*. Luigi Scarso.

16:10 — Pausa caffè (30').

16:40 *Sommari (indici generali) personalizzati*. Gianluca Pignalberi.
17:20 Discussione sulle prospettive del gruppo.

18:00 Chiusura dei lavori. Arrivederci.

La partecipazione all'evento è libera e gratuita.
Registrazione online: www.guitex.org/home/meeting
Informazioni: Gianluca Pignalberi, g.pignalberi@alice.it

Gruppo Utilizzatori Italiani di TEX
GUIT
www.guitex.org

SAPIENZA
UNIVERSITÀ DI ROMA

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 16, Ottobre 2013

Editoriale
Gianluca Pignalberi (?) 5
Scrivere la tesi di laurea in LATEX
Agostino De Marco 6
LATEX e le lingue romanze alpine: il piemontese
Claudio Beccari 30
LATEX e la documentazione di progetto
Roberto Giacomelli 39
Gli alfabeti romani di Francesco Griffo
Claudio Vincoletto 52
Il pacchetto minerva
Grazia Messineo, Salvatore Vassallo 58
Experiments with multitasking and multithreading in LUATEX
Luigi Scarso 68
Sommari (indici generali) personalizzati
Gianluca Pignalberi 84

Gruppo Utilizzatori Italiani di TEX

Editoriale

Gianluca Pignalberi (?)

"Eccoci di nuovo ai saluti finali". Mi sembra di vivere uno di quegli inquietanti *déjà vu* in cui sono alla tastiera del mio computer e attacco l'editoriale scrivendo: "Eccoci di nuovo ai saluti finali". Mi sembra di vivere uno di quegli inquietanti *déjà vu* in cui sono alla tastiera del mio computer e attacco l'editoriale scrivendo: "Eccoci di nuovo ai saluti finali". Mi sembra di vivere uno di quegli inquietanti *déjà vu* in cui sono alla tastiera del mio computer e attacco l'editoriale scrivendo: "Eccoci di nuovo ai saluti finali". Mi sembra di vivere uno di quegli inquietanti *déjà vu* in cui sono alla tastiera del mio computer e attacco l'editoriale scrivendo: "Eccoci di nuovo ai saluti finali". Mi sembra di vivere uno di quegli inquietanti *déjà vu* in cui sono alla tastiera del mio computer e attacco l'editoriale scrivendo: "Eccoci di nuovo ai saluti finali". Mi sembra di vivere uno di quegli inquietanti *déjà vu* in cui sono alla tastiera del mio computer e attacco l'editoriale scrivendo: "Eccoci di nuovo ai saluti finali". Mi sembra di vivere uno di quegli inquietanti *déjà vu* in cui sono alla tastiera del mio computer e attacco l'editoriale scrivendo:

BANG! *Argh!*

Pfffft! Salve a tutti, sono il direttore ombra. Ho appena fatto fuori il vecchio direttore che, detto tra noi, mi stava pure un po' sulle scatole. Ora la direzione di ArsTEXnica è proprio vacante. Se uno di voi non si propone come direttore sarete permanentemente nei guai.

In qualità di ombra sarei autorizzato a sproloquiare dell'operato del fu direttore, quindi lo farò. Era una persona lungimirante, tant'è che ha preceduto la linea del governo Letta + Alfano rimandando a data da destinarsi l'adozione del formato 17 × 24 per ArsTEXnica, formato proposto da Ivan Valbusa e di cui Claudio Beccari ha realizzato la classe. Sempre il fu ha procrastinato all'infinito la scrittura di un manuale di stile per gli autori della nostra rivista. Oltre a procrastinare, il fu era anche un asso nel demandare: demandava spesso a Massimiliano Dominici la realizzazione di pezzetti di software per migliorare la gestione di ArsTEXnica.

Ringrazio io, in sua vece, tutti quanti, dal momento che il fu ha avuto anche il cattivo gusto di farsi togliere dalle balle senza salutare né ringraziare nessuno. Povero mondo!

Parliamo degli articoli, cioè degli interventi al meeting, va'. Cominciamo da Agostino De Marco e dal suo intervento dal titolo *Scrivere la tesi di laurea in LATEX*. L'autore ha raccolto l'impegnativa sfida di portare al meeting quegli studenti e quei tesisti che avranno presto bisogno di LATEX per dare una veste grafica dignitosa ai loro sforzi universitari, magari perché costretti da uno o più professori. Il lavoro, vastissimo, spiega dall'inizio alla fine cosa usare e come usarlo per dare un aspetto professionale alla propria tesi.

Claudio Beccari prosegue la sua opera di diffusione delle lingue romanze alpine, specie nel mondo di LATEX, parlandoci del piemontese. Il suo articolo è, al solito, prodigo di spiegazioni, codice, consigli e osservazioni. Non mancano gli aneddoti personali ad arricchire di umanità un testo scritto da un vero professionista.

Prosegue le presentazioni Roberto Giacomelli. Il suo *LATEX e la documentazione di progetto* si addentra nell'oscuro mondo dei rapporti tecnici. Nel mondo reale la diffusione di LATEX nelle aziende è spesso affidata a singoli volenterosi, consci della sua potenza e versatilità. La maggior parte degli *elaboratori-di-testo-con-Word* trovano una serie infinita di scuse per non capire il miglioramento della qualità del lavoro offerta da LATEX. È probabile che la lettura di questo articolo li convincerebbe di quanto siano in torto.

Claudio Vincoletto ci riporta nel magico mondo dei font parlandoci della sua ultima fatica in *Gli alfabeti romani di Francesco Griffo*. Il suo virtuosismo con METAFONT ci permette di ammirare l'ennesima ottima prova di programmazione di un font condita con ampie spiegazioni storiche e tecniche.

Grazia Messineo e Salvatore Vassallo ci raccontano gli sviluppi del progetto di cui sono parte vitale ne *Il pacchetto* minerva. Il progetto cresce bene come un bambino sano e gli autori forniscono anche l'indirizzo a cui registrarsi per provare il prodotto e il servizio descritto.

Experiments with multitasking and multithreading in LuaTEX di Luigi Scarso ci fa fare il pieno di due cose: storia dei modelli e degli strumenti di elaborazione e la sperimentazione in chiave multitasking e multithreading su Lua e LuaTEX. Ci mostra dunque un possibile scenario futuro sulla composizione tipografica resa più efficiente dall'adozione di strategie computazionali intrinsecamente efficienti.

Infine, de profundis, il fu ci parla delle sue esperienze più o meno lavorative alle prese con la riproduzione dell'aspetto degli indici generali con titletoc e del possibile uso di questo pacchetto anche in un ambiente di produzione e non come mero strumento sperimentale. Riposa in pace.

▷ Gianluca Pignalberi (?)
`g dot pignalberi at alice dot it (?)`

Scrivere la tesi di laurea in LaTeX

Agostino De Marco

Sommario

Lo scopo del presente articolo è fornire gli strumenti per scrivere una tesi di laurea utilizzando LaTeX. Tale obiettivo è conseguito analizzando i problemi tipici incontrati durante la stesura della tesi e le possibili soluzioni; si pone particolare attenzione ai pacchetti da usare nelle varie circostanze. I singoli argomenti non vengono approfonditi nei dettagli ma si rimanda, ove necessario, alla letteratura specifica o ai manuali dei pacchetti suggeriti.

Abstract

The goal of this article is to provide the tools to write a thesis with LaTeX. The article analyzes the problems that are usually encountered while writing a thesis and their solution; a particular emphasis is on the packages to use in each case. The topics are not examined in deep and, when necessary, the reader is referred to specific literature or to the manual of the suggested packages.

1 Introduzione

Questo articolo trae spunto dall'articolo *Scrivere la tesi di laurea con LaTeX 2ε* di Mori (2007). Dal 2007 al 2013 il sistema TeX ha subito aggiornamenti continui — alcuni dei quali hanno segnato degli importanti passi in avanti dal punto di vista tecnologico — e si è arricchito di nuove interessanti funzionalità. Si metteranno qui in risalto gli aggiornamenti più significativi per chi ha intenzione di scrivere la tesi di laurea o la tesi di laurea magistrale, la monografia di laurea o la tesi di dottorato[1] in LaTeX. Come per l'articolo di Mori, il lettore non deve aspettarsi una guida alla redazione della tesi. Per chi voglia saperne di più si rimanda ai testi di Eco (1977), Lesina (2013) e, in particolare per le tesi scientifiche, ai testi di Matricciani (2000), Matricciani (2003), Matricciani (2007) e Beccari *et al.* (2011). Piuttosto, lo scopo dell'articolo è quello di dare delle indicazioni utili e generali per lavorare con LaTeX efficacemente.

Il testo presume che il lettore conosca già i rudimenti di LaTeX, ovvero che abbia letto — o almeno, sia seriamente intenzionato a leggere — una delle numerose guide di base disponibili gratuitamente in rete.

Per i lettori italiani è senz'altro consigliabile consultare *L'arte di scrivere con LaTeX* di Pantieri e Gordini [2] (chiamata in gergo l'*Arte*). Questa fonte è fondamentale soprattutto per i neofiti, che vi troveranno spiegazioni su come procurarsi tutto l'occorrente per usare LaTeX, come installarlo nel proprio calcolatore e come aggiornarne la distribuzione. Inoltre, la guida di Pantieri e Gordini presenta in maniera chiara e organica i concetti fondamentali della composizione tipografica con LaTeX, offrendo un vasto campionario di esempi e di problemi risolti.

Un'alternativa alla guida su menzionata è la *Introduzione all'arte della composizione tipografica con LaTeX* a cura del Gruppo Utilizzatori Italiani di TeX (chiamata in gergo *Guida* GuIt), adatta soprattutto agli utenti desiderosi di approfondire i dettagli del linguaggio LaTeX e i meccanismi della composizione tipografica.

In linea generale in questo articolo si seguirà la prassi di non scandagliare troppo i vari argomenti: dei pacchetti citati, infatti, si analizzano soltanto le impostazioni più importanti e se ne suggerisce l'uso, indirizzando alla relativa documentazione chi voglia approfondirne la conoscenza.

Si ricorda che la maggioranza dei pacchetti per LaTeX è accompagnata da un manuale che ne descrive l'utilizzo e spesso presenta degli esempi. La posizione del manuale dipende dalla distribuzione TeX che si usa; le distribuzioni più diffuse offrono il comando

`texdoc` ⟨*nome pacchetto*⟩

che cerca e apre il file PDF (Portable Document Format) con il manuale del pacchetto indicato. In alternativa, disponendo di un collegamento a internet, è possibile reperire il manuale di un pacchetto all'indirizzo

`http://texdoc.net/pkg/`⟨*nome pacchetto*⟩

oppure si può cercare per parole chiave il documento che interessa attraverso l'interfaccia del sito `http://texdoc.net`.

2 Prescrizioni di formattazione di una tesi di laurea

Tutti gli aspiranti alla laurea si trovano di fronte ad un elenco, più o meno dettagliato, di 'prescrizioni di formattazione' o 'direttive redazionali' per la tesi di laurea. Un tipico esempio potrebbe essere il seguente:

1. Da qui in avanti si userà per brevità la locuzione 'tesi di laurea' o il termine 'tesi' per indicare genericamente questo tipo di documenti.

2. `http://www.lorenzopantieri.net/LaTeX_files/ArteLaTeX.pdf`

> La tesi deve essere composta scrivendo entrambi i lati delle pagine, su fogli di formato UNI A4.
>
> I margini devono essere: superiore 20 mm, inferiore 15 mm, sinistro e destro 15 mm, rilegatura 15 mm.
>
> La distanza dal bordo per intestazione e piè di pagina deve essere di 12,50 mm.
>
> Il carattere da usare è Times New Roman, 11 pt, interlinea doppia.

Oltre a queste specifiche di tipo generale, devono essere precisate le regole di 'stile', cioè il formato delle testatine, dei piedini, dei titolini, eccetera. Molto spesso lo stile del manoscritto viene sottoposto agli studenti attraverso un file preconfezionato (creato con MS Word o con OpenOffice) che fa da modello. Il modello è esso stesso un documento che chiarisce (ma non sempre) i dettagli del layout da adottare.

Si deve osservare, purtroppo, che le prescrizioni di formato nelle università italiane in alcuni casi sono troppo generiche, in altri differiscono a seconda della scuola o dipartimento di appartenenza del relatore della tesi. In altri casi ancora (non rari) le direttive redazionali contengono delle vere e proprie castronerie dal punto di vista della tipografia professionale; verrebbe da pensare che chi ha redatto quelle prescrizioni o non è ancora passato al calcolatore e usa ancora la macchina da scrivere, o usa solamente e male un word processor (che usato bene produrrebbe anche risultati buoni), oppure ignora completamente i rudimenti della tipografia.

Chi intende comporre la tesi in LaTeX ha il compito di interpretare le direttive e scegliere la classe di documento e/o i pacchetti di estensione necessari a raggiungere il risultato voluto. La qualità del risultato dipenderà, ovviamente, dalla predisposizione ad apprendere gli aspetti tecnici e dagli strumenti di cui si è in possesso.

È noto che molti studenti che si avvicinano a LaTeX lo fanno proprio in occasione della stesura della tesi. Per essi è fondamentale un lavoro preparatorio che richiede di:

- installare una distribuzione aggiornata del sistema TeX — ad esempio, la distribuzione TeX Live[3] o la distribuzione MikTeX,[4]
- acquisire dimestichezza con il flusso di lavoro necessario a generare un documento minimale — creazione di un file sorgente, compilazione e creazione di un output in formato PDF,
- comprendere almeno i concetti basilari della tipografia — rudimenti sui font, struttura di un manoscritto, layout, stile, eccetera.

A questo punto sarà possibile cimentarsi con il lavoro di design del manoscritto di laurea.

3. http://www.tug.org/texlive
4. http://miktex.org

3 Classi per le tesi di laurea

Dagli archivi CTAN (Comprehensive TeX Archive Network)[5] si possono scaricare diversi pacchetti che contengono il necessario per comporre la tesi di laurea o di dottorato con LaTeX. Fra i tanti file di estensione, che servono per estendere le classi di documento predefinite alla composizione delle tesi, esistono alcune classi e pacchetti che vale la pena citare: ClassicThesis (Miede, 2013), sapthesis (Biccari, 2012), suftesi (Valbusa, 2013), TOPtesi (Beccari, 2013), frontespizio (Gregorio, 2013), per lo più scritte da italiani per gli studenti universitari italiani.

La classe ClassicThesis scritta da un docente tedesco, ma adatta a tutte le lingue, offre un design della pagina dall'aspetto professionale, che sarebbe quanto mai indesiderabile personalizzare, perché si perderebbe tutto il bello di questo pacchetto. Siccome il layout della pagina è piuttosto originale, può non adattarsi alle specifiche di questa o quella università.

La classe sapthesis offre una soluzione completa per la composizione di tesi per studenti della Sapienza – Università di Roma.[6]

La classe suftesi fornisce uno stile di documento molto semplice e sobrio, vicino alle abitudini estetiche degli utenti umanisti.[7]

Il pacchetto TOPtesi contiene sia il file di classe, sia un pacchetto omonimo con cui si può configurare un certo numero di classi standard, contiene un pacchetto con comandi utili, che può essere usato indipendentemente dalla classe, e un pacchetto per il frontespizio, con il quale probabilmente si possono personalizzare diverse classi, ma non è stato creato come modulo a parte espressamente per questo scopo. La classe consente di comporre tesi in italiano e in inglese: per comporre il frontespizio in lingua diversa dall'italiano si possono usare comandi specifici che possono essere inseriti in un file di configurazione; la tesi è personalizzabile per ogni lingua e per molti stili universitari. La classe è stata pensata anche per scrivere le tesi completamente in lingua diversa dall'italiano in vista del fatto che gli studenti in doppia laurea con i programmi Erasmus devono scrivere la tesi anche (o solo) nella lingua dell'università ospitante.

Merita un'attenzione particolare il pacchetto frontespizio di Enrico Gregorio. Esso si dedica esclusivamente al frontespizio della tesi; questo è completamente configurabile in ogni suo dettaglio, per cui è possibile predisporre il frontespizio della tesi virtualmente per ogni prescrizione di segreteria di

5. http://ctan.org
6. http://biccari.altervista.org/c/informatica/latex/sapthesis.php
7. http://profs.lettere.univr.it/valbusa/2010/09/17/la-classe-suftesi

ateneo.[8] Esempi d'uso di questo pacchetto sono riportati nel paragrafo 6.1.

Per coloro che decidono di comporre il manoscritto con una delle classi su menzionate sarà consigliabile attenersi in primo luogo al manuale della classe scelta. Se si è studenti della Sapienza – Università di Roma e si sceglie sapthesis, oppure, si è studenti del Politecnico di Torino e si sceglie TOPtesi, allora gran parte del lavoro è già predisposto e l'attenzione può essere concentrata con una certa disinvoltura sui contenuti anziché sul layout o sullo stile del documento. Se si ha la necessità di personalizzare queste classi perché si appartiene ad un'altra università e si devono rispettare certe direttive di formato, sarà bene valutare se si è veramente in grado di realizzare le personalizzazioni richieste in un tempo dato. I forum di utilizzatori di LaTeX sono pieni di richieste disperate di aiuto da parte di studenti che hanno poco tempo per la consegna del manoscritto e che non riescono a risolvere questo o quel problema di formattazione.

La classe TOPtesi di Beccari può effettivamente rivelarsi una buona scelta, perché ha un manuale d'uso chiaro ed è abbastanza elastica da poter essere personalizzata anche dai meno esperti.

In alternativa alle soluzioni precedenti si può scegliere di utilizzare la classe di documento predefinita book, selezionando via via i pacchetti di estensione che permettono di realizzare le soluzioni tipografiche desiderate.[9]

La parte rimanente di questo articolo propone appunto questa strada. Naturalmente, a parte la scelta della classe di documento e qualche altro aspetto legato al layout e allo stile, il resto dell'articolo contiene argomenti che interessano anche chi sceglie ClassicThesis, sapthesis, suftesi o TOPtesi o altre classi ancora,[10]

4 La tesi con la classe book

Per una tesi di laurea è possibile utilizzare la classe predefinita book. Nelle opzioni della classe, oltre alla dimensione del font di base (`10pt`, `11pt` o `12pt`)[11] e a quella del foglio (tipicamente `a4paper`), è possibile scegliere:

- se avere un documento fronte-retro (`twoside`) o solo fronte (`oneside`),
- se collocare la prima pagina dei capitoli su facciate destre (`openright`) o indifferentemente (`openany`).

Si suggerisce di utilizzare la classe book invece di quella report in quanto la prima prevede tre comandi (`\frontmatter`, `\mainmatter` e `\backmatter`)[12] che controllano il formato del numero di pagina e la numerazione dei capitoli. Nel *frontmatter* le pagine sono numerate con i numeri romani minuscoli (i, ii, iii, ecc.) ed i capitoli non sono numerati come se si utilizzasse il comando asteriscato `\chapter*{}` ma vanno a finire nell'indice (mentre di solito i capitoli iniziati con `\chapter*{}` non compaiono nell'indice). Nel *mainmatter* le pagine sono numerate con numeri arabi (la numerazione riparte da 1) e i capitoli sono anch'essi numerati con cifre arabe. Nel *backmatter* le pagine sono numerate come nel mainmatter (la numerazione prosegue da questa) ma i capitoli non sono numerati.

Si consiglia inoltre di utilizzare l'opzione fronte-retro (`twoside`) in quanto:

- si dimezza l'uso di fogli di carta,[13]
- è possibile usare testatine differenziate per pagine sinistre e destre,
- i libri sono scritti in questo modo (e dunque ci si aspetta che chi legge la tesi sia abituato a questo layout).

Se ad esempio si vuole avere la tesi con dimensione del corpo 11 pt, stampata fronte-retro su fogli A4, con collocazione della prima pagina dei capitoli su facciate destre, va usato il comando

```
\documentclass[11pt,a4paper,twoside,%
    % ... eventuali altre opzioni
    openright]{book}
```

Alternativamente può essere utilizzata la classe memoir (Wilson, 2010) che risulta particolarmente flessibile e permette di personalizzare molti aspetti del documento (testatine, titoli capitoli, note, indici, ecc.) senza dover caricare altri pacchetti. Si rimanda alla documentazione della classe per i dettagli. L'uso di memoir non è consigliabile per gli utenti neofiti; il manuale d'uso è abbastanza voluminoso e leggerlo e capirlo senza possedere la dovuta predisposizione per la materia potrebbe risultare un carico di lavoro non tollerabile da alcuni. D'altra parte, il manuale di memoir è un'ottima fonte di informazioni per chi vuole approfondire le sue conoscenze sulla tipografia.

8. L'unica cosa a cui bisogna fare attenzione è che il pacchetto frontespizio non consente di riprodurre le prescrizioni di formato di alcune università italiane; ma sono solo casi in cui queste direttive sono inaccettabili dal punto di vista tipografico. Gli studenti universitari che si accingono a scrivere la tesi non si scoraggino: se una cosa non si può fare con frontespizio allora vuol dire che è meglio non farla.

9. In tal caso sarà bene assicurarsi di aver installato una versione completa del sistema TeX così da avere già nel proprio computer i file necessari e pronti all'uso.

10. Ad esempio la classe scrbook del pacchetto KOMA-Script (Kohm e Morawski, 2012).

11. Per avere una buona leggibilità su fogli A4 è consigliabile usare un font di base di dimensione 11 pt.

12. Per l'uso di tali comandi si rimanda al paragrafo 6.

13. Un comportamento comune a molti laureandi consiste nell'usare qualunque strumento tipografico possibile per aumentare il numero di pagine della tesi (allargando i margini, aumentando la dimensione del font, aumentando l'interlinea, inserendo molte figure, stampando solo fronte, ecc.). Tralasciando il fatto che la qualità dei contenuti è più importante della quantità, spesso questi espedienti producono dei risultati tipografici pessimi. Si consiglia dunque di concentrarsi sui contenuti e lasciar perdere l'impostazione tipografica (a questo pensa LaTeX) ed in particolare il numero di pagine prodotto.

5 Organizzazione dei file

5.1 La codifica dei sorgenti

Il problema della codifica dei file di testo è delicato e spesso difficile da capire per chi non conosce il funzionamento interno del proprio calcolatore. Per approfondire l'argomento si consiglia la guida di Beccari e Gordini (2012).

Dal punto di vista pratico gli utenti di LaTeX devono preoccuparsi di come il proprio editor[14] gestisce la codifica dei caratteri. Se ne cita qui uno per tutti: TeXworks,[15] che è l'editor multipiattaforma che si installa quando viene installata la distribuzione TeX Live o la distribuzione MikTeX. Si dice *codifica di input* il modo in cui sono codificati i caratteri che si immettono nei file `.tex` (e nei file di testo in generale), vuoi attraverso tastiera ed editor, vuoi leggendo con quest'ultimo un file preesistente per modificarlo. Normalmente un editor salva i file con la stessa codifica di quella con cui è configurato. L'editor TeXworks può anche salvarli con una codifica diversa da quella di default. Di solito questa funzionalità non è un problema, anzi può essere molto utile per cambiare codifica a un file.

Si consiglia di impostare la codifica dei file sorgenti della tesi come UTF-8. Per informare il programma di composizione sulla codifica con cui il file sorgente è salvato, basta mettere nel preambolo la chiamata al pacchetto inputenc, specificando nel suo argomento la sigla della codifica in questione. In pratica, nel preambolo basta dare il comando:

```
\usepackage[utf8]{inputenc}
```

Prima di caricare inputenc si consiglia di caricare non solo il pacchetto fontenc con le opzioni che si desiderano, ma anche il pacchetto textcomp e ogni altro pacchetto che carichi collezioni di simboli speciali, nell'ordine seguente:

```
\usepackage[T1]{fontenc}
% ... eventuali pacch. per font particolari
\usepackage{textcomp}
% ... eventuali pacch. per simboli speciali
\usepackage[utf8]{inputenc}
```

L'editor TeXworks ha il vantaggio di comprendere particolari istruzioni di autoconfigurazione sulla base delle quali adattare 'al volo' le proprie impostazioni, qualunque esse siano. Un utente di TeXworks ha la possibilità di 'configurare il sorgente' immettendo all'inizio del documento delle righe di commento speciali, anche dette in gergo *righe magiche*. Le righe magiche danno a TeXworks alcune importanti informazioni di autoconfigurazione:

- se esiste e come si chiama il file principale (un'ovvietà superflua se il documento è in un unico file, ma molto utile se si suddividono i sorgenti di un lungo documento in più file);
- la codifica usata per scriverlo (ad esempio, `UTF-8 Unicode`);
- il programma di composizione che si userà per comporlo (ad esempio, `pdflatex`, `xelatex` o `lualatex`);
- volendo, il dizionario ortografico della lingua principale del documento.

Ecco come vanno scritte queste righe (gli spazi resi qui con il simbolo ␣ sono significativi):

```
%␣!TEX␣root␣=␣./tesi.tex
%␣!TEX␣encoding␣=␣UTF-8␣Unicode
%␣!TEX␣program␣=␣pdflatex
%␣!TEX␣spellcheck␣=␣it-IT
```

Ciò fatto, anche se si lavora su computer e sistemi operativi diversi, usando l'editor impostato con codifiche diverse (ad esempio, su uno è impostata di default la codifica latin1 e sull'altro la UTF-8), si può aprire il file in questione e lavorarci sopra senza dover usare alcuna accortezza preliminare, poiché sarà l'editor ad autoconfigurarsi correttamente.

5.2 Suddivisione dei sorgenti

La gestione di documenti articolati come un libro o una tesi di laurea può diventare complessa e dunque è auspicabile suddividere il testo in più file. LaTeX permette di avere un *main file* che viene compilato per produrre il risultato finale ed in cui sono richiamati altri file sorgenti. Molti usano nominare il file principale `main.tex`; per una tesi di laurea si potrebbe scegliere il nome `tesi.tex`. Eventualmente, se la tesi deve essere consegnata come file PDF, l'output della compilazione può essere rinominato da `tesi.pdf` a `Tesi_Magistrale_Matteo_Rossi.pdf`, per esempio.

Gli altri file sorgenti vengono richiamati dal sorgente principale con i comandi `\include` e `\input`. Il comando

```
\input{⟨nome file⟩}
```

permette il *nesting*, ovvero rende possibile richiamare un file che ne richiama a sua volta un altro. Il comando

```
\include{⟨nome file⟩}
```

non permette il *nesting*, ma inserisce un comando `\clearpage` prima del testo che contiene e permette di utilizzare il comando

```
\includeonly{⟨nome file 1⟩,
   ⟨nome file 2⟩, ⟨nome file 3⟩ ... }
```

per inserire solo i file specificati tra parentesi. Quando si usa `\includeonly` vengono compilati solamente i file tra parentesi graffe e si aggiornano i

14. Programma di creazione e di gestione dei file di testo, cioè dei sorgenti.

15. `http://www.tug.org/texworks`. Si può scaricare un'agile introduzione al programma da `http://profs.sci.univr.it/~gregorio/introtexworks.pdf`.

contatori ad essi collegati (numeri di pagina, numeri di note, ecc.). I contatori dei file già compilati e non inclusi da `\includeonly` non vengono aggiornati.

6 Sezioni della tesi

L'organizzazione della tesi di laurea è argomento di specifici manuali di scrittura (Eco, 1977; Lesina, 2013; Matricciani, 2000, 2003) ed in particolar modo della normativa ISO relativa alla presentazione dei rapporti scientifici e tecnici UNI-ISO 5966 (1989). Molto dettagliata è la guida di Beccari *et al.* (2011) liberamente scaricabile dal sito del Politecnico di Torino. In questo paragrafo si propone una possibile struttura per la tesi e si affrontano le problematiche relative ad ogni sezione.

Una tesi può in generale presentarsi con la seguente struttura:[16]

- Il frontespizio°
- La dedica*°
- Il sommario*°
- I ringraziamenti*°
- Gli indici°
- I simboli e le notazioni*
- La prefazione*

} frontmatter

- I capitoli interni
- Le appendici*

} mainmatter

- La bibliografia
- L'elenco degli acronimi*
- L'indice analitico*

} backmatter

6.1 Il frontespizio

La struttura ed il contenuto del frontespizio sono generalmente imposti dalla scuola presso cui la laurea è conseguita, dunque è necessario crearlo *ad hoc*. Uno dei problemi che spesso si presentano è quello di produrre un frontespizio adeguato che sia ben centrato sulla prima pagina.

Per gli utenti italiani esiste una soluzione già pronta e facilmente personalizzabile, costituita dal pacchetto frontespizio. Il vantaggio di usare questo pacchetto è che i comandi necessari per definire i vari elementi del frontespizio (titolo, candidato, relatore e così via) sono contenuti nello stesso documento.

Per definire il frontespizio si deve usare l'ambiente `frontespizio` che può essere posizionato subito dopo il comando `\begin{document}`. All'interno dell'ambiente vanno dati i comandi che definiscono i vari elementi del frontespizio. Se il documento principale si chiama `tesi.tex`, alla prima compilazione verrà generato automaticamente il documento `tesi-frn.tex`, che si troverà nella stessa cartella che contiene quello principale. Il documento `tesi-frn.tex` va anch'esso compilato per generare il file `tesi-frn.pdf`, che verrà posizionato automaticamente come prima pagina di `tesi.pdf`. La sequenza di comandi è, dunque,

```
pdflatex tesi
pdflatex tesi-frn
pdflatex tesi
```

e, alla fine, il frontespizio sarà al suo posto. Non occorrerà dare ogni volta questi comandi: basta farlo solo quando si modifica il contenuto dell'ambiente `frontespizio`.

Se la classe book è chiamata con l'opzione `oneside`, il frontespizio occupa correttamente solo la prima pagina; nel caso di `twoside`, viene prodotta una seconda pagina bianca.

Il documento va impostato dando al comando `\documentclass` l'opzione `titlepage` per poi caricare nel preambolo il pacchetto frontespizio. Per esempio:

```
\documentclass[a4paper,
  % ... altre opzioni
  titlepage]{book}
% ... altri comandi del preambolo
\usepackage{frontespizio}

\begin{document}
\begin{frontespizio}
\Universita{Padova}
\Facolta{Scienze Matematiche, Fisiche e
    Naturali}
\Corso[Laurea]{Matematica}
\Titoletto{Tesi di laurea}
\Titolo{Equivalenze fra categorie di moduli\\
e applicazioni}
\Candidato[145822]{Enrico Gregorio}
\Relatore{Ch.mo Prof.~Adalberto Orsatti}
\Annoaccademico{2012-2013}
\end{frontespizio}
% ... il resto della tesi
\end{document}
```

produce il frontespizio riportato nella figura 1a. L'esempio seguente:

```
\documentclass[a4paper,titlepage]{book}
\usepackage[swapnames]{frontespizio}
\begin{document}
\begin{frontespizio}
\begin{Preambolo*}
\usepackage{fourier}
```

16. Il simbolo * contraddistingue le sezioni facoltative mentre ° indica che le sezioni non devono essere presenti nell'indice.

(a) (b)

Figura 1: Esempi di frontespizio.

```
\newcommand{\VOF}{\textsc{vof}}
\end{Preambolo*}
\Universita{Napoli Federico II}
\Logo[2.5cm]{Sigillo_UNINA_FedericoII_BLUE}
\Dipartimento{Ingegneria Industriale}
\Corso[Dottorato di Ricerca]{Ingegneria
    Industriale}
\Titolo{Optimization of volume of fluid
  (\VOF) methods\\
  and two-phase flows simulations
}
\Candidato{Agostino~De~Marco}
\Relatore{Ch.mo Prof.~Ermanno Lanconelli}
\NRelatore{Coordinatore}{}
\Correlatore{Ch.mo Prof.~Adalberto Orsatti}
\NCorrelatore{Supervisore della ricerca}{}
\Annoaccademico{2012-2013}
\end{frontespizio}
...
\end{document}
```

produce il frontespizio della figura 1b e mostra anche la possibilità di inserire un'immagine che rappresenta il logo dell'ateneo.

6.2 La dedica

La dedica, ove presente, può assumere le più svariate forme a seconda dei gusti dell'autore. Di solito (vedi ad esempio la figura 2) è costituita da una riga allineata a destra ad esempio con i comandi

```
\begin{flushright}
...
\end{flushright}
```

La posizione verticale della riga nella pagina può essere scelta a piacere e per controllarla risulta particolarmente conveniente l'uso di una coppia di comandi `\vspace{\stretch{...}}`. In questo modo è infatti possibile impostare il rapporto tra lo spazio che precede la dedica e quello che segue. Se ad esempio si vuole che lo spazio che segue sia il doppio di quello che precede, è possibile usare i comandi

```
\null\vspace{\stretch{1}}
\begin{flushright}
  \textit{A Valeria e ai miei genitori}
\end{flushright}
\vspace{\stretch{2}}\null
```

6.3 Il sommario

Le classi article e report — ma non di default la classe book — definiscono un ambiente

```
\begin{abstract}
...
\end{abstract}
```

per il sommario o *abstract* dei contenuti di un documento. Se si utilizza la classe book è necessario inserire nel preambolo la definizione di tale ambien-

FIGURA 2: Esempio di dedica.

te. Si riporta qui una definizione ispirata a quella della classe report

```
nel preambolo
\usepackage{fancyhdr}

\newenvironment{abstract}%
  {\cleardoublepage%
    \thispagestyle{empty}%
    \null \vfill\begin{center}%
      \bfseries \abstractname \end{center}}%
  {\vfill\null}
```

Si noti l'utilizzo del pacchetto fancyhdr al quale si accennerà nel paragrafo 9.2.1.

Per le tesi di laurea in italiano è spesso richiesto che sia presente anche la traduzione inglese dell'abstract. Utilizzando il pacchetto babel è possibile selezionare la lingua per le due versioni dell'abstract in modo che sia effettuata la corretta sillabazione delle parole e che sia caricato automaticamente il corretto titolo del sommario. Dopo aver richiamato il pacchetto nel preambolo con il comando

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

è sufficiente inserire i sommari come segue

```
\begin{abstract}
... versione del sommario in italiano ...
\end{abstract}

\selectlanguage{english}
\begin{abstract}
... English version of the abstract ...
\end{abstract}
\selectlanguage{italian}
```

Il risultato è riportato nella figura 3.

6.4 Gli indici

Gli indici di solito sono posizionati subito dopo il sommario nel seguente ordine:

- indice
- elenco delle figure
- elenco delle tabelle
- altri elenchi

e vengono prodotti automaticamente da LaTeX con i comandi

```
\tableofcontents
\listoffigures
\listoftables
```

Per creare elenchi di oggetti flottanti personalizzati (ad esempio listati di programmi, algoritmi, eccetera) si faccia riferimento al pacchetto float ed ai relativi comandi `\newfloat` e `\listof`. Per modificare il layout degli indici è possibile utilizzare il pacchetto tocloft.

6.5 I simboli e le notazioni

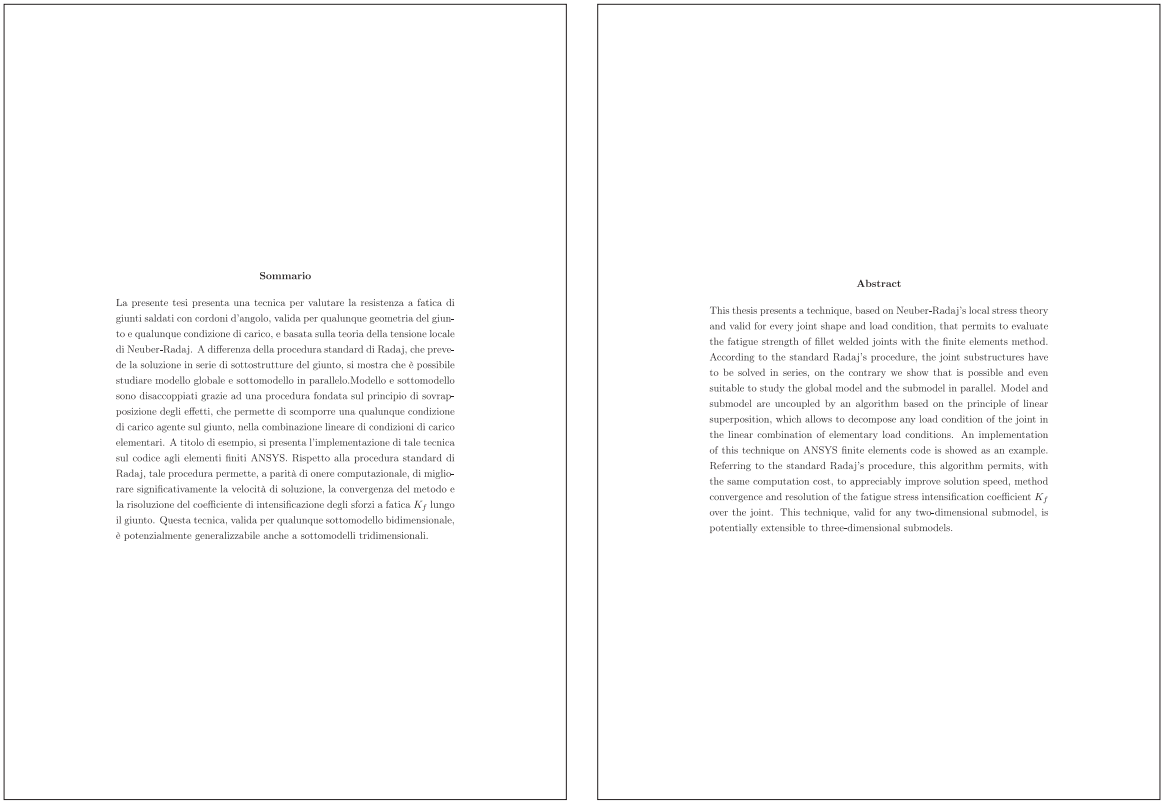
Talvolta risulta opportuno far precedere al testo della tesi un elenco dei simboli e delle notazioni utilizzate. A questo scopo può essere utilizzato il pacchetto nomencl. Un'alternativa più potente a nomencl è il pacchetto glossaries che permette anche di creare un elenco degli acronimi menzionati nel testo e un glossario. Entrambi i pacchetti generano gli elenchi automaticamente per mezzo del programma `makeindex` e, accoppiati all'uso del pacchetto hyperref, generano automaticamente anche i collegamenti ipertestuali tra il simbolo, l'acronimo, il termine menzionato nel testo e la relativa spiegazione nell'elenco. Si rimanda ai rispettivi manuali d'uso per approfondimenti.

Ovviamente, per semplicità, è anche possibile creare manualmente l'elenco, ad esempio utilizzando l'ambiente tabular. Nella figura 4 si riporta un esempio.

6.6 Le appendici

Le appendici sono dei normali capitoli la cui numerazione è però in lettere latine. LaTeX permette di crearle semplicemente con il comando `\chapter{...}` preceduto da `\appendix`; se si hanno più appendici, `\appendix` deve essere richiamato solo una volta. Si riporta un esempio:

```
...
\mainmatter
\include{capitolo1}
\include{capitolo2}
\include{capitolo3}
```



Sommario

La presente tesi presenta una tecnica per valutare la resistenza a fatica di giunti saldati con cordoni d'angolo, valida per qualunque geometria del giunto e qualunque condizione di carico, e basata sulla teoria della tensione locale di Neuber-Radaj. A differenza della procedura standard di Radaj, che prevede la soluzione in serie di sottostrutture del giunto, si mostra che è possibile studiare modello globale e sottomodello in parallelo.Modello e sottomodello sono disaccoppiati grazie ad una procedura fondata sul principio di sovrapposizione degli effetti, che permette di scomporre una qualunque condizione di carico agente sul giunto, nella combinazione lineare di condizioni di carico elementari. A titolo di esempio, si presenta l'implementazione di tale tecnica sul codice agli elementi finiti ANSYS. Rispetto alla procedura standard di Radaj, tale procedura permette, a parità di onere computazionale, di migliorare significativamente la velocità di soluzione, la convergenza del metodo e la risoluzione del coefficiente di intensificazione degli sforzi a fatica K_f lungo il giunto. Questa tecnica, valida per qualunque sottomodello bidimensionale, è potenzialmente generalizzabile anche a sottomodelli tridimensionali.

Abstract

This thesis presents a technique, based on Neuber-Radaj's local stress theory and valid for every joint shape and load condition, that permits to evaluate the fatigue strength of fillet welded joints with the finite elements method. According to the standard Radaj's procedure, the joint substructures have to be solved in series, on the contrary we show that is possible and even suitable to study the global model and the submodel in parallel. Model and submodel are uncoupled by an algorithm based on the principle of linear superposition, which allows to decompose any load condition of the joint in the linear combination of elementary load conditions. An implementation of this technique on ANSYS finite elements code is showed as an example. Referring to the standard Radaj's procedure, this algorithm permits, with the same computation cost, to appreciably improve solution speed, method convergence and resolution of the fatigue stress intensification coefficient K_f over the joint. This technique, valid for any two-dimensional submodel, is potentially extensible to three-dimensional submodels.

(a) (b)

FIGURA 3: Esempio di abstract in doppia lingua.

```
\appendix
\include{appendice1}
\include{appendice2}
...
```

6.7 L'indice analitico

L'indice analitico può essere creato automaticamente per mezzo del pacchetto imakeidx.

L'esempio seguente:

```
\usepackage{imakeidx}
...
\makeindex[title=Concept index]
\makeindex[name=persons,title=Index of
    names,columns=3]
...
\begin{document}
...
la relatività.\index{relativity}
...
Einstein.\index[persone]{Einstein, Albert}
...
E fu da quel punto che fu data alla teoria il
nome di \emph{Teoria della relatività}.

\printindex

\indexprologue{\small
  In questo indice troverete un elenco
  di scienziati famosi citati in questa
  tesi.
}
\printindex[persone]

\end{document}
```

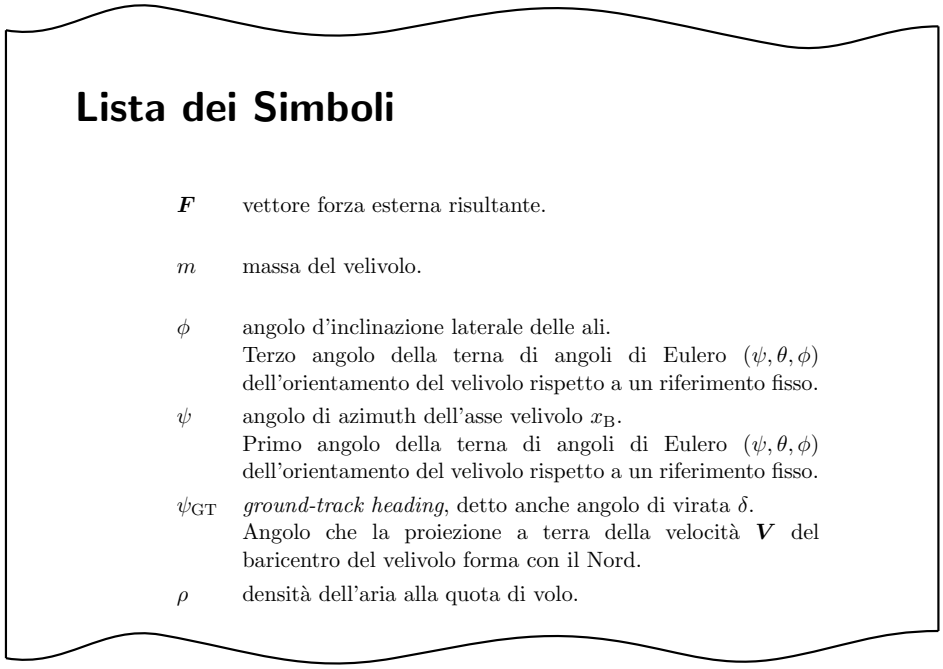
produce due indici analitici. Il secondo è preceduto da un breve testo di spiegazione.

Si rimanda al manuale d'uso del pacchetto per le eventuali necessità di personalizzazione del formato.

6.8 La bibliografia

La bibliografia è una parte importante della tesi di laurea. LATEX offre tutti gli strumenti per realizzarla e gestirla con efficienza e flessibilità. L'argomento richiede la comprensione di alcuni aspetti tecnici e, al solito, si consiglia di approfondirne i dettagli consultando la guida di PANTIERI e GORDINI. Qui si richiamano gli elementi fondamentali per gestire le citazioni bibliografiche e il database delle fonti con il pacchetto biblatex.

La gestione efficiente della bibliografia è basata sulla generazione automatica di un insieme di voci bibliografiche citate durante il testo della tesi. Le voci bibliografiche vengono 'estratte' da una collezione (database) di fonti preparata in precedenza. Il database è un file di testo di estensione `.bib` che va editato a parte inserendovi dei record opportunamente formattati. Esiste un ottimo programma multipiattaforma per la creazione di



Lista dei Simboli

$\boldsymbol{F}$	vettore forza esterna risultante.
m	massa del velivolo.
ϕ	angolo d'inclinazione laterale delle ali. Terzo angolo della terna di angoli di Eulero (ψ, θ, ϕ) dell'orientamento del velivolo rispetto a un riferimento fisso.
ψ	angolo di azimuth dell'asse velivolo x_{B}. Primo angolo della terna di angoli di Eulero (ψ, θ, ϕ) dell'orientamento del velivolo rispetto a un riferimento fisso.
ψ_{GT}	*ground-track heading*, detto anche angolo di virata δ. Angolo che la proiezione a terra della velocità $\boldsymbol{V}$ del baricentro del velivolo forma con il Nord.
ρ	densità dell'aria alla quota di volo.

FIGURA 4: Esempio di elenco dei simboli.

database bibliografici chiamato Jabref.[17] Esso è dotato di un'interfaccia grafica e di potenti funzioni di gestione.

Un esempio di database bibliografico contenente un certo numero di record è il seguente:

```
@book{eco:tesi,
  author = {Eco, Umberto},
  title = {Come si fa una tesi di laurea},
  publisher = {Bompiani},
  date = {1977},
  location = {Milano},
}

@article{mori:tesi,
  author = {Mori, Lapo Filippo},
  title = {Scrivere la tesi di laurea con
    \LaTeX},
  journaltitle = {\Ars},
  number = {3},
  date = {2007},
}

@manual{beccari:gordini:codifiche,
  title = {Codifiche in {\TeX} e {\LaTeX}.
    Dal sorgente al PDF, guida pratica per
    lavorare con successo.},
  author = {Beccari, Claudio and Gordini,
    Tommaso},
  publisher = {{\GuIT}},
  year = {2012},
}

@online{wiki:latex,
  title = {\LaTeX{} su Wikipedia},
  date = {2012},
  url = {http://it.wikipedia.org/wiki/LaTeX},
  sortkey = {wiki},
  label = {wiki},
}
```

Il primo record è un esempio di voce bibliografica riferita a un libro (`@book`), il secondo è un esempio di articolo su rivista (`@article`). il terzo record è un riferimento a un manuale (`@manual`), il quarto è un sito internet (`@online`).

Ciascun record ha dei campi che vanno dal titolo, all'autore, all'anno di pubblicazione, e così via.[18] I record sono identificati da una *chiave*; per esempio il record del libro di Eco ha per chiave `eco:tesi`. Le chiavi vengono stabilite dall'utente e devono essere usate nel testo della tesi allorquando si vuole inserire una citazione bibliografica.

Il programma 'estrattore' delle voci bibliografiche dal file `.bib`, che lavora tenendo conto delle effettive citazioni presenti nella tesi, si chiama `biber` e fa parte delle moderne distribuzioni TEX.

Il pacchetto biblatex è un potentissimo strumento — pensato per interfacciarsi con `biber` — con il quale si gestisce automaticamente la bibliografia e si personalizza ogni aspetto degli stili bibliografici e di citazione con poche operazioni. Per funzionare correttamente, il pacchetto richiede di caricare anche il pacchetto babel (o polyglossia, se si compone con XELATEX) e csquotes con le opzioni indicate di seguito.

Nel preambolo vanno dati i comandi:

17. `http://jabref.sourceforge.net`

18. Ciascun campo, come si vede, va terminato con la virgola, *anche se è l'ultimo*, pena un errore.

```
\usepackage[italian]{babel}% tesi in italiano

\usepackage[
    autostyle,italian=guillemets
    % ... altre opzioni
  ]{csquotes}

\usepackage[
    % ... opzioni
    backend=biber
  ]{biblatex}
```

Se, per esempio, la base di dati è stata nominata `bibliografia-tesi.bib`, per indicare a LaTeX di usare questo file per comporre la bibliografia, si deve dare nel preambolo il comando

```
\addbibresource{bibliografia-tesi.bib}
```

Se si ha più di un database bibliografico, il comando precedente deve essere ripetuto per ogni file e specificando sempre l'estensione `.bib`.

A questo punto, nel testo della tesi, la citazione di una fonte bibliografica sarà semplicemente ottenuta con un qualcosa di simile:

```
Si veda~\cite{eco:tesi} per maggiori
    dettagli.
```

cioè con il comando `\cite`. Per ottenere il risultato voluto, che potrebbe essere il seguente:

> Si veda [1] per maggiori dettagli.

la sequenza di comandi di compilazione è:

```
pdflatex tesi
biber tesi
pdflatex tesi
pdflatex tesi
```

Ovviamente questa sequenza è richiesta solo quando si modifica `bibliografia-tesi.bib` (per esempio, dopo aver aggiunto una voce o aver corretto un errore). La doppia compilazione con `pdflatex` dopo l'esecuzione di `biber` è necessaria per la corretta gestione delle informazioni trascritte nei file ausiliari.

Si osservi che il formato finale della citazione dipende dallo stile richiesto tramite le opzioni passate al pacchetto biblatex. L'esempio precedente potrebbe avere l'aspetto seguente:

> Si veda ECO (1977) per maggiori dettagli.

in cui lo stile della citazione è passato da quello 'numerico' a quello cosiddetto 'autore-anno'.

Il comando `\printbibliography` posizionato al termine del testo della tesi produce la *sezione bibliografica* con relativi titolo e testatina. La sezione bibliografica non è altro che l'elenco dei riferimenti bibliografici, opportunamente ordinati e formattati.

Per mandare nell'indice generale il titolo della bibliografia del documento, va data la sequenza di comandi seguente (valida per la classe di documento `book`):

```
\addcontentsline{toc}{chapter}{\bibname}
\printbibliography
```

Se si usa `babel` per un documento in italiano il comando `\bibname` produce nell'indice generale del documento la voce *Bibliografia*. Se nell'indice si vuole la voce *Riferimenti bibliografici* basta ridefinire `\bibname` tramite il comando

```
\addto\captionsitalian{%
  \renewcommand*{\bibname}%
    {Riferimenti bibliografici}%
}
```

Si osservi che l'aspetto dei riferimenti bibliografici e delle citazioni, che biblatex adatta automaticamente alla lingua principale del documento, si specificano in diversi modi. Il pacchetto fornisce quattro stili bibliografici predefiniti, i quali agiscono nella sezione bibliografica del documento. Essi ordinano le opere (ad esempio, alfabeticamente in base al cognome di autore o curatore); possono contrassegnare o meno l'opera con un'etichetta; sistemano opportunamente i dati nei riferimenti bibliografici.

Si rimanda all'*Arte* di Pantieri e Gordini o al manuale di biblatex (Lehman, 2013) per approfondimenti sugli stili e sugli schemi di citazione.

7 Gli oggetti

7.1 Le figure

Le figure sono uno degli argomenti trattati più estesamente dalle guide. Al solito, l'*Arte* di Pantieri e Gordini è un'ottima fonte di approfondimento.

I problemi incontrati dagli utenti LaTeX durante l'inserimento di figure sono generalmente di due tipi. Una parte dei problemi derivano dalle figure in sé, ovvero dal file che si cerca di inserire in un documento (verrà trattato nel par. 7.1.1), mentre un altro tipo di problemi, totalmente distinto dal precedente, è quello degli oggetti flottanti (e verrà trattato nel par. 7.3).

7.1.1 Formati

Esistono due grandi classi di figure, le immagini vettoriali e le immagini bitmap. Le prime sono descritte da forme e possono essere scalate e/o deformate senza perdere definizione; sono soprattutto adatte per i grafici e per gli schemi. Le seconde sono matrici di pixel colorati e sono adatte per le fotografie.

La prima cosa da fare è produrre figure nel formato più adatto per i propri scopi. È inutile salvare grafici o schemi in `.jpeg` per poi convertirli in `.pdf`, in quanto la conversione di un'immagine bitmap in `.pdf` include semplicemente il file bitmap in una "cornice" (tipica del formato Encapsulated PostScript, da cui il PDF deriva) senza migliorare in alcun modo la qualità. È inutile anche fare la

conversione opposta, da file vettoriale a bitmap, perché in questo modo si perdono le informazioni sulla geometria contenuta nella figura e quindi si abbassa la qualità del file.

L'elemento più importante di un file PDF è il Bounding Box (BB), che determina la taglia effettiva dell'immagine e che serve a LaTeX per calcolare lo spazio da riservare alla figura. Idealmente i BB dovrebbero essere al limite massimo del contenuto dell'immagine, ma spesso i programmi di grafica lasciano grandi bordi bianchi attorno alla figura disegnata. Questo porta spesso a grandi confusioni, perché di fatto LaTeX sta lasciando alla figura lo spazio corretto, ma visivamente parte di questo è utilizzato per il bordo bianco, quindi la figura appare troppo piccola, non centrata, con eccessivi margini verticali, eccetera. La prima cosa da verificare è quindi che il programma di grafica generi dei file `.pdf` con BB corretti. Per farlo basta aprire la figura con Ghostview[19] e attivare la visualizzazione dei BB. Se questi non sono corretti bisogna cercare di configurare correttamente il programma di grafica, ma il problema non ha niente a che vedere con LaTeX. Nel caso si abbiano molti file che presentano questo inconveniente bisogna cercare di correggere il problema all'origine.

7.1.2 Pacchetti utili

Per inserire le figure è necessario caricare il pacchetto graphicx, della cui guida si consiglia la lettura. Per ottenere sottofigure (vedi ad esempio la figura 1) è necessario caricare il pacchetto subcaption.

In casi semplici non è necessario ricorrere a quest'ultimo pacchetto visto che all'interno degli ambienti figure e table si può mettere più di un grafico o di una tabella. Se si hanno quindi due o più figure che possono essere raggruppate insieme, scrivendo

```
\begin{figure}[tb]
  \begin{minipage}{0.48\textwidth}
    \includegraphics[%
      width=\linewidth]{fig_a.pdf}
    \caption{Prima didascalia (destra).}
    \label{fig:a}
  \end{minipage}
  \hspace{4em}
  \begin{minipage}{0.48\textwidth}
    \includegraphics[%
      width=\linewidth]{fig_b.pdf}
    \caption{Seconda didascalia (sinistra).}
    \label{fig:b}
  \end{minipage}
\end{figure}
```

In questo modo si riduce il numero di oggetti flottanti e se ne facilita l'inserimento.

Al fine di mantenere ordine nei file sorgenti, è consigliabile raccogliere tutte le figure in una o più sottocartelle; se ad esempio tali sottocartelle si chiamano `dir_1` e `dir_2`, è sufficiente inserire nel preambolo con il seguente comando del pacchetto graphicx:

```
\graphicspath{{dir_1/},{dir_2/}}
```

L'argomento di `\graphicspath` è relativo alla cartella dove risiede il main file `.tex` che viene compilato.

La formattazione delle didascalie può essere convenientemente controllata con il pacchetto caption.

Qui vale la pena di menzionare il pacchetto di estensione adjustbox che, tra le sue svariate funzionalità, offre il comando `\adjincludegraphics` (simile a `\includegraphics`) che permette di effettuare agevoli operazioni di rifilatura (*cropping*). Ad esempio il codice

```
\adjincludegraphics[width=0.7\linewidth,
  trim={{.05\width} {.02\height} 0 0},% lbrt
  clip]{mia-figura.pdf}
```

inserisce l'immagine `mia-figura.pdf` ritagliandone dal lato sinistro (`l`, *left*) una striscia di larghezza pari al 5% della larghezza originale e dal lato in basso (`b`, *bottom*) una striscia di altezza uguale a 2% dell'altezza originale. Il risultato del ritaglio viene poi scalato in modo da avere un'immagine sulla pagina di larghezza uguale al 70% della `\linewidth`.

7.2 Le tabelle

Così come per le figure, anche per le tabelle esistono guide specifiche a cui si rimanda per ogni approfondimento (Mori, 2006).

Per migliorare la spaziatura dell'ambiente tabular standard è possibile utilizzare il pacchetto ctable, mentre se si vogliono colorare le righe o le colonne è necessario caricare il pacchetto xcolor con l'opzione `table`.

Nel caso di tabelle di grandi dimensioni è possibile ridurre la dimensione della tabella effettuando una scalatura, ad esempio con i seguenti comandi:

```
\begin{center}
  \resizebox{0.95\textwidth}{!}{%
    \begin{tabular}
      ...
    \end{tabular}
  }
\end{center}
```

Si può anche ruotare la tabella di 90° con il pacchetto rotating. Altre tecniche per ruotare immagini e tabelle sono indicate nel manuale del pacchetto hvfloat (Voss, 2013).

Infine, si può spezzare la tabella su più pagine con il versatile pacchetto longtable.

7.3 Controllo degli oggetti flottanti

Spesso gli utenti si lamentano del fatto che LaTeX sposti le figure (e in generale gli oggetti flottanti) lontano dal punto in cui vengono inserite. Nella

19. `http://pages.cs.wisc.edu/~ghost`

maggioranza dei casi questo è dovuto ad un utilizzo erroneo delle opzioni di posizionamento degli oggetti flottanti. Qui si vuole sottolineare che alcune scelte devono essere prese nella fase di stesura del testo (paragrafo 7.3.1) mentre altre sono riservate, quando necessarie, alla fase di revisione (paragrafo 7.3.2).

7.3.1 Cosa fare durante la stesura del testo

In primo luogo bisogna accettare il fatto che se LaTeX sposta un oggetto flottante è perché lo spazio è fisicamente insufficiente, o per motivi estetico-tipografici. Per esempio LaTeX non metterà mai una figura seguita da un titolo di sezione e da un cambio pagina, ma preferirà stampare la sezione e poi la figura, oppure se si aggiunge un oggetto flottante in fondo ad una pagina, LaTeX è obbligato a spostarlo almeno nella pagina successiva. Se lo spazio è insufficiente, è inutile cercare di forzare LaTeX a mettere l'oggetto flottante in tale posizione: se lo spazio fisico non c'è, non si può certo inventarlo.

Per fortuna con un minimo di accortezza LaTeX fa un ottimo lavoro. Per prima cosa è opportuno utilizzare sempre il posizionamento automatico evitando di aggiungere `\clearpage` o comandi simili: in fase di redazione chi scrive la tesi dovrebbe solo concentrarsi sui contenuti e non sull'impaginazione. In generale i posizionamenti fatti a mano interferiscono con la complessa routine di LaTeX per il posizionamento degli oggetti flottanti e portano a risultati peggiori rispetto a quelli di default. Seguendo le semplici indicazioni che seguono, il posizionamento automatico mantiene gli oggetti flottanti vicini al punto di inserimento ed inoltre evita che l'utente si preoccupi continuamente del posizionamento dei float, lasciando più tempo per lavorare sui contenuti.

Una delle origini dei problemi lamentati è l'utilizzo eccessivo dell'opzione `[h]` (che chiede di posizionare la figura nel punto dove compare nel codice): gli oggetti flottanti vengono spesso inseriti con l'opzione `[htbp]` o peggio `[h!t]`. In generale si pensa che questa opzione sia la migliore per mantenere gli oggetti flottanti vicino al punto di inserimento. In realtà può funzionare bene solo quando gli oggetti inseriti sono molto piccoli (dove per piccolo si intende con un'altezza molto inferiore rispetto all'altezza del corpo del testo). Il modo migliore per utilizzare le opzioni di posizionamento è quello di domandarsi in primo luogo se l'oggetto flottante sarà abbastanza piccolo per stare in una pagina di testo o se avrà bisogno di una pagina tutta per sé. Nel primo caso lo si introduce quindi con un'opzione di posizionamento `[tb]`, nel secondo con `[p]`. Se non ci sono oggetti flottanti in sospeso, nel primo caso LaTeX potrà spostare l'oggetto subito prima del punto di inserzione (cosa che non può fare se si usa `[h]`) o nella pagina immediatamente successiva. Usando invece `[p]` per i grossi oggetti flottanti, questi verranno immediatamente stampati in una pagina dedicata, e non verrano spostati alla fine del capitolo come succede con `[tbp]`. Basta sfogliare un qualunque testo ben impaginato per accorgersi che le figure sono introdotte proprio in questo modo: in generale all'inizio o alla fine della pagina, in una pagina intera se sono grandi, raramente nel corpo del testo se sono davvero piccole. Alcuni utenti sono infastiditi dal fatto che alcuni oggetti flottanti appaiano prima del testo in cui sono citati (ad esempio una figura in alto nella pagina in cui è citata): per risolvere questo problema è possibile utilizzare il pacchetto flafter che impedisce agli oggetti flottanti di apparire prima della loro definizione nel testo.

Infine è utile ricordare che LaTeX riesce a posizionare tutte le figure in modo corretto solo se il rapporto

$$\frac{\text{testo}}{\text{figure}}$$

è sufficientemente alto. Da questo segue che è auspicabile (per altro non solo per fini tipografici) scrivere qualche cosa di interessante piuttosto che riempire le lacune con immagini. Se tale rapporto è troppo basso, può accadere che la compilazione si interrompa e venga restituito il seguente errore:

```
! LaTeX Error: Too many unprocessed floats.
```

Questo è dovuto al fatto che LaTeX ha una certa quantità di memoria dedicata al posizionamento degli oggetti flottanti; se troppi oggetti si accumulano durante la compilazione tale memoria può esaurirsi. Per risolvere questo problema è possibile utilizzare il pacchetto placeins. Esso definisce il comando `\FloatBarrier` che non può essere oltrepassato dagli oggetti flottanti e quindi impone il posizionamento di tutti quelli che sono ancora in memoria. Nel caso che il documento presenti dei posti dove possa essere inserita un'interruzione di pagina, conviene utilizzare `\clearpage`. Tale comando, oltre a creare un'interruzione di pagina, impone il posizionamento di tutti gli oggetti flottanti ancora in memoria in modo analogo a `\FloatBarrier`. Il pacchetto morefloats aumenta il numero di oggetti flottanti che possono essere mantenuti in memoria durante la compilazione da 18 a 36.

Se tutto ciò non fosse sufficiente, nella fase precedente la stampa, e solamente allora, è possibile intervenire manualmente come spiegato nel paragrafo seguente.

7.3.2 Cosa fare durante la revisione del testo

Nella fase che precede la stampa può essere necessario intervenire manualmente per correggere il posizionamento degli oggetti flottanti (quali ad esempio le figure e le tabelle). A questo riguardo esistono numerosi pacchetti, di cui i più utili sono costituiti da float e placeins.

Il pacchetto float permette di forzare il posizionamento dell'oggetto nel punto in cui è situato il relativo ambiente per mezzo dell'opzione `H`. A volte è utile usare questa opzione insieme al comando `\afterpage` del pacchetto afterpage.

Il pacchetto placeins permette di mettere delle barriere invalicabili per gli oggetti flottanti con il comando `\FloatBarrier`.

Il motore di composizione TEX mette a disposizione parametri che controllano gli oggetti flottanti:

`\setcounter{topnumber}{...}` massimo numero di float in posizione `t` per ogni pagina

`\def\topfraction{...}` massima frazione di pagina per i float in posizione `t` per ogni pagina

`\setcounter{bottomnumber}{...}` massimo numero di float in posizione `b` per ogni pagina

`\def\bottomfraction{...}` massima frazione di pagina per i float in posizione `b` per ogni pagina

`\setcounter{totalnumber}{...}` massimo numero di float nella stessa pagina

`\setcounter{dbltopnumber}{...}` massimo numero di float grandi nella stessa pagina

`\def\textfraction{...}` minima frazione di pagina per il testo

`\def\floatpagefraction{...}` minima frazione di pagina per i float in posizione `p`

`\def\dbltopfraction{...}` massima frazione di pagina per i float a piena pagina in composizione a due colonne in posizione `t`

`\def\dblfloatpagefraction{...}` minima frazione di pagina per i float a piena pagina in composizione a due colonne in posizione `p`

Va osservato che l'intervento manuale per forzare il posizionamento degli oggetti flottanti può portare a risultati rovinosi se non si è compreso appieno il meccanismo standard di svuotamento delle code dei float. I comandi su elencati — soprattutto l'opzione `H` per gli ambienti `figure` e `table` e il comando `\floatbarrier` — devono essere usati con estrema cautela.

8 Compilare il codice

8.1 PDF come formato di output

Fino a qualche anno fa il codice LATEX doveva essere compilato per ottenere in output un file in formato DeVice-Independent (`.dvi`); successivamente si otteneva un file in formato PDF per conversione di formato. Questo schema di lavoro non è più usato.

Oggi, con le moderne distribuzioni di TEX, la compilazione attraverso il programma `pdflatex` (`xelatex` o `lualatex`) produce direttamente un file in formato PDF (`.pdf`).

Un altro vantaggio delle distribuzioni moderne è la possibilità di effettuare la cosiddetta *ricerca diretta*[20] e la *ricerca inversa*,[21] molto utili in fase di elaborazione della tesi.

8.2 Il formato PDF archiviabile

La norma ISO 19005-1 del 2005 stabilisce il formato di archiviazione come un formato derivato dal formato PDF mediante alcune aggiunte e modifiche al normale formato PDF, tanto che questo formato di archiviazione si chiama PDF/A.

La tesi, quindi, non potrebbe essere consegnata al momento dell'iscrizione all'esame di laurea in un formato qualsiasi, sia esso DOC, ODT, PS, RTF, o altri formati più o meno esoterici, liberi o proprietari; nemmeno il formato PDF di per sé ha il formato giusto, se manca delle altre piccole modifiche e aggiunte a cui si accennava sopra. Anche il formato PDF scelto per la tesi dovrebbe corrispondere alla versione PDF-1.4 e non dovrebbero essere accettabili né versioni precedenti né versioni successive, perché così prescrive la norma ISO. Le piccole modifiche e aggiunte possono venire inserite su di un file in formato PDF-1.4 mediante opportuni applicativi ancora non molto diffusi e, in particolare, ancora per lo più commerciali.

Il programma `pdflatex` (a partire dagli aggiornamenti del 2008) è in grado di generare direttamente file in formato PDF/A mediante l'ausilio di un file di estensione contenuto nel pacchetto pdfx scaricabile dagli archivi CTAN. Su qualunque piattaforma, oltre al pacchetto pdfx, bisogna caricare anche i file del modello di colore; questi file si possono installare direttamente nella cartella dove si sono installati tutti i file del pacchetto pdfx, in particolare dove si è installato `pdfx.sty`. Il pacchetto può creare sia file conformi allo standard PDF/A sia a quello PDF/X.

Per approfondire questo argomento è vivamente consigliata la consultazione della *Guida* GUIT (GRUPPO UTILIZZATORI ITALIANI DI TEX, 2013).

9 Pacchetti utili

9.1 La lingua italiana

9.1.1 Norme tipografiche

In italiano la maggioranza delle regole tipografiche non sono universali e vincolanti, ma dipendono piuttosto da convenzioni e abitudini o dal gusto dell'autore. Nonostante questo, è importante che l'autore della tesi conosca quali sono le principali "norme" tipografiche italiane. CEVOLANI (2006) ne

20. facendo CTRL+click sul codice all'interno dell'editor TEXworks, la finestra di visualizzazione del `.pdf` scorre fino a trovare il rispettivo output.

21. facendo CTRL+click all'interno della finestra di visualizzazione del `.pdf`, il cursore viene posizionato sul rispettivo codice all'interno dell'editor

offre una sintesi e, per ogni regola, mostra come applicarla in LaTeX.

9.1.2 La sillabazione

Per attivare la sillabazione italiana e caricare i nomi delle sezioni[22] in lingua italiana, è necessario caricare il pacchetto babel con l'opzione per l'italiano per mezzo del comando

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

Ecco il tipico inizio di un sorgente per un documento in italiano con la corretta sequenza dei pacchetti da caricare:

```
\documentclass[11pt,a4paper,twoside,%
    % ... eventuali altre opzioni
    openright]{book}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[italian]{babel}
```

Il pacchetto babel definisce alcuni comandi molto utili per trattare correttamente ciascuna lingua in un documento multilingue. Supponendo di dover scrivere un documento in italiano con alcune parti in inglese, babel andrà caricato così:

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

Per singole parole o brevi frasi in lingua straniera è disponibile il comando `foreignlanguage`. Eccone un esempio:

```
\foreignlanguage{english}{
   This text in in English!
}
```

Per porzioni di testo in lingua più consistenti è disponibile l'ambiente `otherlanguage` come nell'esempio seguente:

```
\begin{otherlanguage*}{english}
  This is a very long text in English. It
    should be hyphenated correctly.
\end{otherlanguage*}
```

LaTeX 2ε sillaba correttamente quasi tutte le parole italiane, tuttavia esistono casi in cui si utilizzano nomi propri oppure parole rare; in questa eventualità, se la sillabazione tentata da LaTeX 2ε non è soddisfacente, è possibile suggerirla con il comando `\hyphenation` (va posizionato nel preambolo): si devono scrivere le parole sillabate tra parentesi graffe, separate da uno spazio, come nel seguente esempio:

```
\hyphenation{sil-la-ba-zio-ne sim-pa-ti-ca}
```

Il precedente comando può anche essere utilizzato quando si vuole che alcune parole *non* vengano sillabate: è sufficiente scriverle senza trattini come nel seguente esempio

22. Ad esempio "sommario", "bibliografia", "indice", eccetera.

```
\hyphenation{MATLAB Mathematica}
```

Tale comando può anche essere utilizzato per forzare una sillabazione particolare; se ad esempio si vuole che la parola "melograno" sia spezzata tra "melo" e "grano" e non in altri punti, è sufficiente scrivere:

```
\hyphenation{melo-grano}
```

Se la parola in questione compare una sola volta, è possibile suggerirne la sillabazione direttamente nel testo con `\-`; avremo ad esempio

```
sil\-la\-ba\-zio\-ne
```

Vale la pena di segnalare che con l'opzione `italian` di babel il carattere `"` è attivo e serve per svolgere una serie di funzioni, tra le quali:

- introdurre una possibile cesura disabilitando le altre cesure "troppo vicine" ma consentendo la sillabazione di entrambi i monconi della parola; per esempio `dispepsia` viene divisa naturalmente in `di-spep-sia` mentre `dis"pepsia` viene divisa in `dis-pep-sia`;
- andare a capo negli URL e nelle parole composte in cui i componenti sono separati da una barra; `input"/output` spontaneamente non sarebbe divisibile ma con il segno `"` lo diventa, e lo diventano i singoli monconi.

A conclusione di questo paragrafo, è doveroso ricordare che gli interventi manuali sulla sillabazione dovrebbero sempre essere fatti nella fase di revisione che precede immediatamente la stampa. Spesso è preferibile riformulare una frase che dà luogo ad un errore di `overfull` piuttosto che imporre particolari sillabazioni.

9.1.3 Il rientro della prima riga

I libri italiani contemporanei generalmente non hanno il primo capoverso dopo il titoletto di sezione rientrato, tuttavia alcuni autori preferiscono avere tale rientro. Per attivare il rientro sulla prima riga di ogni sezione, sottosezione, eccetera, è necessario caricare il pacchetto indentfirst, in quanto la convenzione anglosassone (di default su LaTeX) non lo prevede. Si veda la figura 5.

9.1.4 Caratteri accentati

In LaTeX i caratteri accentati possono essere introdotti con i comandi standard `\'{e}`, `\'{e}`, eccetera, oppure direttamente da tastiera è, é, e così via, se nel preambolo si carica il pacchetto inputenc con la codifica appropriata. Si consiglia di caricare il pacchetto con l'opzione `[utf8]` (si veda il paragrafo 5.1).

9.2 Il layout

9.2.1 Le testatine ed i piè di pagina

Per personalizzare testatine e piè di pagina è possibile usare il pacchetto fancyhdr. Per una tesi è probabile che si abbiano impostazioni differenti

Problema

Il metodo appena presentato nel par. 2.5.1 presenta un serio problema di fondo. Esso è infatti valido nell'ipotesi che l'insieme di vettori $\hat{\Pi}$ costituisca una base di $\mathbb{R}^6$, ovvero che essi siano linearmente indipendenti. Tuttavia è facile dimostrare che $\det \hat{\mathbf{\Pi}} = 0$ indipendentemente dai valori assegnati alle sei coppie di elementi non nulli della matrice. Conseguenza immediata è che la matrice $\hat{\mathbf{\Pi}}$ non è invertibile e che dunque non è possibile risolvere il problema con questa via.

Conclusioni

Interessante è individuare l'origine della lineare dipendenza dei vettori di $\hat{\Pi}$. Come già osservato, l'insieme di vettori Π è un buon candidato a divenire base di $\mathbb{R}^6$, purché nella scelta dei 12 elementi non nulli si rispetti la condizione (2.4).

Tale condizione viene dunque a mancare nel momento in cui si impongono i vincoli (2.5); in altre parole, il metodo viene a fallire quando si impone che le condizioni di carico di base siano una ad una equilibrate. D'altronde il vincolo di condizioni di carico di base equilibrate è imprescindibile dato che altrimenti risulta impossibile risolvere con metodi numerici agli EF il sottomodello. È dunque necessario trovare un'altra strada per risolvere il problema della scomposizione della condizione di carico generica applicata al giunto Φ.

(a) Senza pacchetto indentfirst.

Problema

Il metodo appena presentato nel par. 2.5.1 presenta un serio problema di fondo. Esso è infatti valido nell'ipotesi che l'insieme di vettori $\hat{\Pi}$ costituisca una base di $\mathbb{R}^6$, ovvero che essi siano linearmente indipendenti. Tuttavia è facile dimostrare che $\det \hat{\mathbf{\Pi}} = 0$ indipendentemente dai valori assegnati alle sei coppie di elementi non nulli della matrice. Conseguenza immediata è che la matrice $\hat{\mathbf{\Pi}}$ non è invertibile e che dunque non è possibile risolvere il problema con questa via.

Conclusioni

Interessante è individuare l'origine della lineare dipendenza dei vettori di $\hat{\Pi}$. Come già osservato, l'insieme di vettori Π è un buon candidato a divenire base di $\mathbb{R}^6$, purché nella scelta dei 12 elementi non nulli si rispetti la condizione (2.4).

Tale condizione viene dunque a mancare nel momento in cui si impongono i vincoli (2.5); in altre parole, il metodo viene a fallire quando si impone che le condizioni di carico di base siano una ad una equilibrate. D'altronde il vincolo di condizioni di carico di base equilibrate è imprescindibile dato che altrimenti risulta impossibile risolvere con metodi numerici agli EF il sottomodello. È dunque necessario trovare un'altra strada per risolvere il problema della scomposizione della condizione di carico generica applicata al giunto Φ.

(b) Con pacchetto indentfirst.

Figura 5: Rientro sulla prima riga.

a seconda della sezione e dunque è conveniente definire alcuni comandi personalizzati che modifichino testatine e piè di pagina; un esempio per `frontmatter` e `mainmatter` potrebbe essere:

```
\newcommand{\fncyfront}{%
    \fancyhead[RO]{{\footnotesize\rightmark}}
    \fancyfoot[RO]{\thepage}
    \fancyhead[LE]{\footnotesize{\leftmark}}
    \fancyfoot[LE]{\thepage}
    \fancyhead[RE,LO]{}
    \fancyfoot[C]{}
    \renewcommand{\headrulewidth}{0.3pt}}
\newcommand{\fncymain}{%
    \fancyhead[RO]{{\footnotesize\rightmark}}
    \fancyfoot[RO]{\thepage}
    \fancyhead[LE]{{\footnotesize\leftmark}}
    \fancyfoot[LE]{\thepage}
    \fancyfoot[C]{}
    \renewcommand{\headrulewidth}{0.3pt}}
```

da utilizzare nel seguente modo:

```
\pagestyle{fancy}
\fncyfront
\frontmatter
...
\fncymain
\mainmatter
```

La definizione di tali comandi è differente a seconda che il testo sia fronte-retro (`twoside`) o solo fronte (`oneside`).

Utilizzando l'opzione `openright` può capitare di ottenere una pagina bianca alla fine di un capitolo; per evitare che in questa pagina siano presenti testatine o piè di pagina, è sufficiente includere nel preambolo il pacchetto emptypage.

In alternativa a fancyhdr è possibile usare titlesec. Tale pacchetto ha un'interfaccia d'uso leggermente diversa da fancyhdr e permette di definire stili diversi da applicarsi in diverse parti del documento.

9.2.2 Il layout della pagina

Molto di frequente i regolamenti degli atenei richiedono un layout della pagina differente da quello prodotto di default dalle classi di LaTeX ed è dunque necessario modificarlo. Il primo modo per intervenire è l'utilizzo di comandi interni del LaTeX, quali `\textwidth`, `\oddsidemargin`, eccetera, tuttavia questa strada è sconsigliabile per molte ragioni.

Una migliore soluzione è costituita dal pacchetto geometry (Umeki, 2010) che è completamente configurabile. Nel caso che siano necessari degli interventi locali a pagine o a paragrafi è possibile utilizzare il pacchetto changepage.

Per rilegare la tesi può essere conveniente indicare sulle pagine dove tagliare il foglio; questo può essere agevolmente realizzato utilizzando in coppia i pacchetti geometry e crop. Si veda ad esempio la figura 6.

Di default LaTeX cerca di coprire interamente con il testo o altri elementi l'intera altezza della pagina e, ove necessario, inserisce degli spazi aggiuntivi tra i capoversi oppure dilata gli spazi tra le voci degli elenchi puntati e così via. Se si vuole disattivare questa impostazione ed avere dello spazio bianco a piè di pagina quando non si riesce a coprirla tutta, è sufficiente aggiungere nel preambolo il comando `\raggedbottom`. Il comportamento di default è invece dovuto al comando `\flushbottom`. Per migliorare la copertura delle pagine è possibile permettere che siano spezzate le formule matematiche in *display* aggiungendo al preambolo il comando `\allowdisplaybreaks` (che funziona solo se è stato caricato il pacchetto amsmath).

È conveniente non modificare il comportamento di default di LaTeX fino a quando non si arriva alla versione definitiva del testo (che precede immediatamente la stampa). Solo in questa fase è possibile intervenire modificando il posizionamento degli oggetti flottanti (vedi il paragrafo 7.3), oppure intervenendo con i comandi appena citati. Prima di modificare le impostazioni di LaTeX è conveniente provare ad effettuare piccole modifiche al testo che spesso sono sufficienti per risolvere i problemi e permettono di ottenere layout più eleganti.

Per approfondimenti sui layout di pagina si consiglia la guida *Introduzione alla definizione della geometria della pagina* (Beccari, 2012).

9.2.3 L'interlinea

Spesso le prescrizioni redazionali impongono un'interlinea diversa da 1 (valore di default in LaTeX). Per modificare l'interlinea del documento esistono più strade, tuttavia la più indicata consiste nel caricare il pacchetto setspace. Tale pacchetto fornisce tre interlinee predefinite richiamate con i comandi `\singlespacing` (interlinea singola), `\onehalfspacing` (interlinea 1,5) e `\doublespacing` (interlinea doppia). Se è necessaria un'interlinea differente, è sufficiente utilizzare il comando `\linespread{...}` mettendo tra parentesi graffe il numero che rappresenta il fattore di scala per l'avanzamento di riga.

9.3 Lo stile

9.3.1 I fonts

In primo luogo, lavorando con `pdflatex`, è consigliabile utilizzare l'encoding T1 che rappresenta lo standard di codifica dei caratteri di LaTeX. Tale codifica è attivata nel preambolo per mezzo del comando

```
\usepackage[T1]{fontenc}
```

Se la tesi è di tipo scientifico, è conveniente abilitare i font matematici forniti dall'$\mathcal{AMS}$ (American Mathematical Society) con il comando

```
\usepackage{amssymb}
```

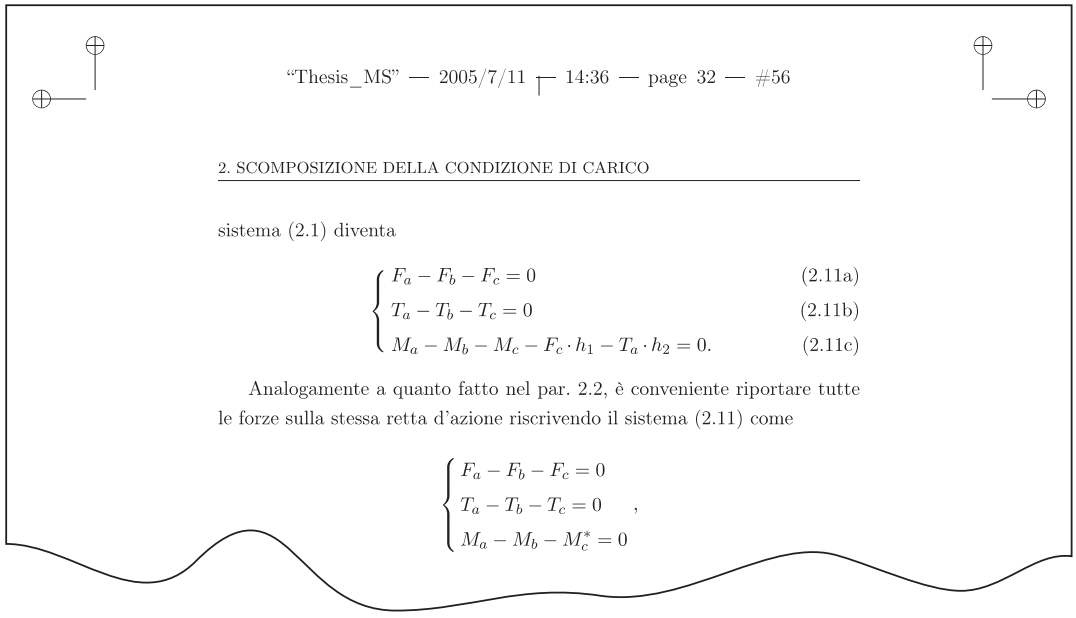


FIGURA 6: Esempio di segni per il taglio del foglio.

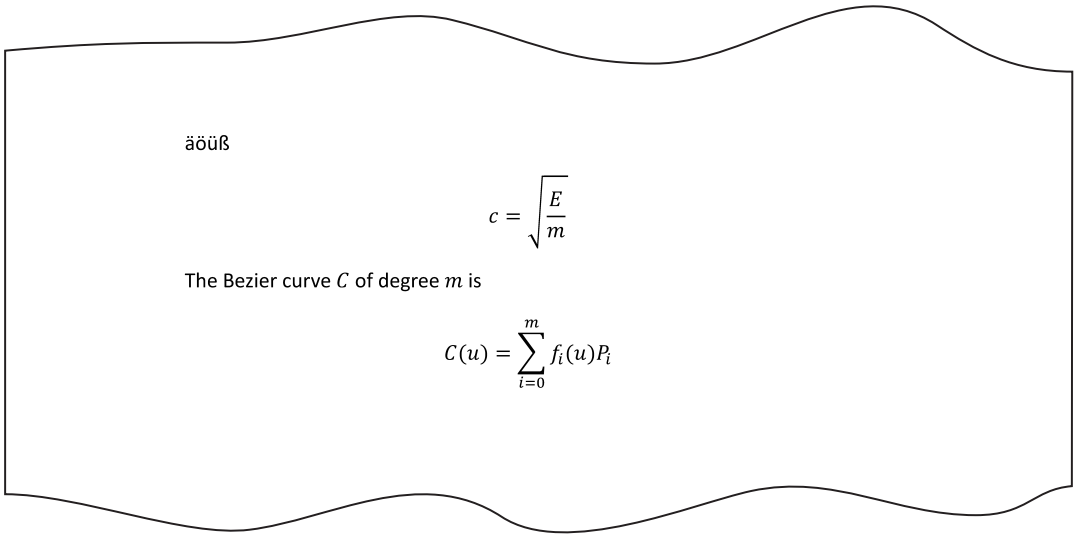


FIGURA 7: Esempio d'uso dei font Calibri per il testo e Cambria Math per la matematica.

Il pacchetto amssymb carica il pacchetto amsfonts e definisce i comandi per usare i simboli introdotti da quest'ultimo.

Per la matematica conviene in generale aggiungere il comando

```
\usepackage{mathtools}
```

che carica anche il pacchetto amsmath e fornisce svariate estensioni per il miglioramento della struttura informativa e della stampa di documenti che contengono formule matematiche.

Per modificare la dimensione del font, in aggiunta ai comandi standard,[23] è utile il pacchetto relsize che consente di assegnare dimensioni relative con i comandi `\smaller` e `\larger`.

Riguardo al tipo di font da utilizzare, l'esperienza conferma che, quasi certamente, la scelta migliore è quella di usare i font che LaTeX carica di default, ovvero la famiglia Computer Modern sviluppata dallo stesso inventore del TeX, Donald Knuth. Questi caratteri possono essere usati nella variante Latin Modern prodotto dal GUST (gruppo utenti di TeX polacco)[24] caricando il pacchetto lmodern:

```
\usepackage[T1]{fontenc}
\usepackage{lmodern}
```

Se si vuole a tutti i costi cambiare font, è bene ricordare che è necessario scegliere quattro famiglie (Serif, Sans-serif, Typewriter e i font per la matematica) che formino una buona combinazio-

23. `\tiny`, `\scriptsize`, `\footnotesize`, `\small`, `\normalsize`, `\large`, `\Large`, `\LARGE`, `\huge` e `\Huge`.

24. `http://www.gust.org.pl/projects/e-foundry/latin-modern`

ne. A tale proposito, è importante ricordare che i font, tranne alcune eccezioni,[25] non hanno tutti i simboli necessari per la matematica e quindi non possono essere usati se non nel testo.

Per cambiare font è possibile caricare uno dei numerosi pacchetti dedicati. Un elenco dei font disponibili e i pacchetti da caricare per usarli è riportato sul sito del gruppo di utenti TEX danese.[26]

Un'ottima alternativa ai caratteri di default è offerta dai pacchetti newtxtext e newtxmath. Con essi si caricano: il font Times (o un suo clone) per il testo, un font appositamente disegnato per la matematica basato su Times Italic, un clone del font Helvetica per la famiglia sans serif, vari possibili font per la famiglia typewriter.

Parlando qui di font, non si può fare a meno di menzionare un'importante alternativa al programma pdflatex, cioè xelatex o anche XeLATEX. La caratteristica principale di XeLATEX è che può adoperare senza bisogno di installazioni particolari tutti i font noti al sistema operativo in uso, che siano in formato OpenType o TrueType. Questi font sono dotati di tabelle interne con cui XeLATEX è capace di creare al volo la struttura dati che nel TEX tradizionale risiede in particolari file metrici (.tfm). Altra importante caratteristica di XeLATEX è che lavora direttamente con file in codifica Unicode, cioè UTF-8 oppure UTF-16. Un esempio di sorgente con caratteri speciali è il seguente:

```
\documentclass{article}

\usepackage{fontspec}
\usepackage{unicode-math}
\setmainfont{Calibri}
\setmathfont{Cambria Math}

\begin{document}
äöüß
\[
c = \sqrt{\frac{E}{m}}
\]
The Bezier curve $C$ of degree $m$ is
\[
C(u) = \sum_{i=0}^m f_i(u)P_i
\]
\end{document}
```

Questo sorgente produce un output come quello della figura 7 in cui si nota l'uso del font Calibri per il testo e del font Cambria Math per la matematica.

Per una introduzione all'uso di XeLATEX si rimanda alla guida di Gregorio (2011) e ai riferimenti in essa contenuti.

9.3.2 Il titolo dei capitoli

Per personalizzare il formato dei titoli dei capitoli è possibile utilizzare il pacchetto fncychap; si rimanda a Mori (2007) per maggiori dettagli.

Nella figura 8 è mostrato un esempio di personalizzazione attraverso il pacchetto titlesec. Una volta definito lo stile nel preambolo con il comando \titleformat la formattazione del titolo del capitolo è ottenuta semplicemente nel documento attraverso il comando standard della classe book:

```
\chapter{Definizioni di base e notazioni}
```

9.3.3 Liste

Per personalizzare i tre ambienti standard dedicati alle liste, cioè enumerate, itemize e description, ci consiglia il pacchetto enumitem.

9.3.4 I "mini indici"

Quando i capitoli hanno una struttura particolarmente complessa, può essere conveniente riportare nella pagina iniziale l'indice del capitolo (vedi ad esempio la figura 9). Questi "mini indici" possono essere prodotti automaticamente con il pacchetto minitoc.

9.3.5 Le epigrafi

Talvolta si vogliono inserire epigrafi nella pagina iniziale dei capitoli. Per farlo è possibile utilizzare il pacchetto epigraph; un esempio è riportato nella figura 10.

9.3.6 Le note

LATEX produce di default un layout delle note di alta qualità; esistono tuttavia alcuni accorgimenti per modificarlo, quando lo si ritenga strettamente necessario. Il pacchetto footmisc fornisce molti controlli sulle note tra cui la possibilità di forzare le note al fondo della pagina[27] con l'opzione bottom; si veda la figura 11.

Per impedire che le note vengano spezzate su più pagine è sufficiente assegnare al parametro di penalità un valore molto elevato, ad esempio

```
\interfootnotelinepenalty=10000
```

mentre per controllare la dimensione della zona assegnata alle note a piè di pagina si può usare il comando

```
\dimen\footins=2cm
```

9.4 La matematica

Si consiglia il lettore di consultare l'*Arte* di Pantieri e Gordini, la guida del Gruppo Utilizzatori Italiani di TEX (2013) e la guida di Voss (2010) per approfondimenti sulla scrittura matematica (semplice e avanzata) in LATEX. Qui di seguito si accenna ad alcuni aspetti interessanti per chi scrive una tesi di laurea.

25. Il gruppo utenti TEX danese ospita una pagina in cui sono riportati tutti i font che supportano la matematica: http://www.tug.dk/FontCatalogue/mathfonts.html.

26. http://www.tug.dk/FontCatalogue

27. Normalmente LATEX unisce le note con l'ultima riga della pagina e dunque su pagine non piene non si hanno le note a fondo pagina.

```
\usepackage[calcwidth,pagestyles]{titlesec}% loads titleps
\usepackage{adjustbox,xcolor}

% chapter head style via titlesec
\titleformat{\chapter}[display]
    {\bfseries\Large}
    {\color{blue!65!black}\filleft%
        \minsizebox{!}{24pt}{\chaptertitlename}% needs package adjustbox
        \lapbox[0pt]{\width}{%
            \minsizebox{!}{40pt}{%
                \ \colorbox{blue!65!black}{\color{white}\thechapter}% needs xcolor
            }%
        }% needs package adjustbox
    }
    {4ex}
    {{\color{blue!65!black}\titlerule}
        \huge\bfseries\scshape
        \vspace{2ex}%
        \filright}
    [\vspace{2ex}%
        {\color{blue!65!black}\titlerule}]
```

Capitolo 1

DEFINIZIONI DI BASE E NOTAZIONI

Jesce sole, jesce sole, nun ce fa' cchiù suspirà!
– Gatta Cenerentola

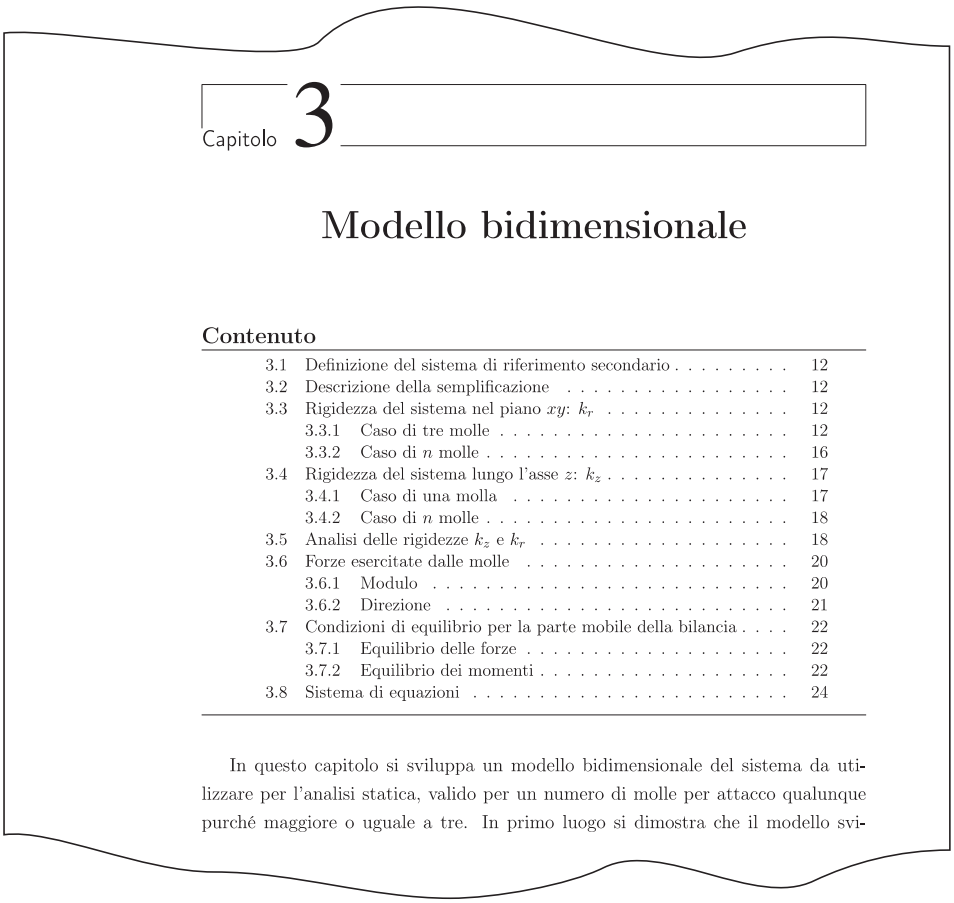
1.1 Introduzione

Lo studio della Meccanica del volo, come altre materie ingegneristiche, poggia le sue basi sui noti concetti della Fisica matematica. Esso richiede di familiarizzare con un certo numero di definizioni, con precise convenzioni sul segno di determinate grandezze e con il sistema di notazione che da esse scaturisce. Più avanti si vedrà che una peculiarità del sistema di notazione della Meccanica del volo, e in particolare dell'Aerodinamica degli aeromobili, è quella di fare largo uso di simboli con pedici multipli.

Scopo di questo capitolo è quello di richiamare i principali elementi di base della materia, a partire dalla definizione dei sistemi di riferimento essenziali e dell'orientamento dei velivoli nello spazio, per passare poi dall'anatomia dei velivoli tradizionali con una panoramica sulle azioni esterne agenti sugli aeromobili in volo. Sarà presentato al tempo stesso il sistema di notazioni adottato nel testo illustrando le motivazioni per cui si scelgono determinati simboli, pedici, eccetera.

Come in tutte le materie ingegneristiche, per le quantità che verranno via via introdotte si utilizzeranno sistemi di unità di misura diversi a seconda del contesto e dell'argomento. Al giorno d'oggi è necessario esprimere le grandezze nel Sistema Internazionale di unità di misura (SI, *International System of Units*). In Italia ne è stato reso obbligatorio l'uso nel 1976 in tutti gli atti pubblici. In Inghilterra e negli USA non vi è alcun obbligo a non utilizzare i sistemi tradizionali di misura basati sulle *Imperial units* e sulle *United States customary units* (o *English Units*). Pertanto, per ragioni storiche, oltre che pratiche, in aeronautica si utilizzano indifferentemente le unità di questi diversi sistemi. Così verrà fatto anche qui.

FIGURA 8: Esempio di definizione del titolo dei capitoli con titlesec. Il comando standard `\chapter` della classe book produce un titolo nel formato personalizzato.



Capitolo 3

Modello bidimensionale

Contenuto

3.1 Definizione del sistema di riferimento secondario . . . 12
3.2 Descrizione della semplificazione . . . 12
3.3 Rigidezza del sistema nel piano xy: k_r . . . 12
3.3.1 Caso di tre molle . . . 12
3.3.2 Caso di n molle . . . 16
3.4 Rigidezza del sistema lungo l'asse z: k_z . . . 17
3.4.1 Caso di una molla . . . 17
3.4.2 Caso di n molle . . . 18
3.5 Analisi delle rigidezze k_z e k_r . . . 18
3.6 Forze esercitate dalle molle . . . 20
3.6.1 Modulo . . . 20
3.6.2 Direzione . . . 21
3.7 Condizioni di equilibrio per la parte mobile della bilancia . . . 22
3.7.1 Equilibrio delle forze . . . 22
3.7.2 Equilibrio dei momenti . . . 22
3.8 Sistema di equazioni . . . 24

In questo capitolo si sviluppa un modello bidimensionale del sistema da utilizzare per l'analisi statica, valido per un numero di molle per attacco qualunque purché maggiore o uguale a tre. In primo luogo si dimostra che il modello svi-

FIGURA 9: Esempio di "mini indice".

9.4.1 I simboli "speciali"

Intendendo con "simboli speciali" tutti quelli che non sono inseribili direttamente dalla tastiera, è necessario distinguere tra quelli matematici e quelli non matematici: per i primi dovrebbe essere sufficiente caricare i simboli dell'$\mathcal{AMS}$ con il pacchetto amssymb; per tutti gli altri simboli sono necessari pacchetti appositi che possono essere facilmente identificati consultando la preziosa guida *The comprehensive LATEX symbol list* (PAKIN).

9.4.2 Rappresentazione dei numeri

Un pacchetto molto utile per la rappresentazione di numeri è numprint. Tra le funzioni di tale pacchetto si ricordano l'inserimento di un separatore ogni tre cifre per le migliaia e l'approssimazione automatica. Ad esempio

```
\numprint{2.742647826672E-01}
```

produce

$$2{,}743 \cdot 10^{-01}$$

9.4.3 Unità di misura

Le unità di misura del Sistema Internazionale possono essere inserite con i comandi del pacchetto siunitx (se ne veda la documentazione), che permette di regolarne molto finemente il formato e di cambiare il risultato nel documento finito operando un'unica modifica nel preambolo anziché agire a mano su ciascuna unità di misura. Dato che le convenzioni tipografiche italiane prevedono la virgola e non il punto (predefinito dal pacchetto) come separatore decimale, il pacchetto va caricato almeno con l'opzione seguente:

```
\usepackage[output-decimal-marker={,}
  % ... altre opzioni
  ]{siunitx}
```

I comandi fondamentali sono `\num` (simile a `numprint`), `\SI` e `\si` che permettono di formattare i numeri, le grandezze fisiche e le unità di misura in maniera configurabile.

Il pacchetto dispone di un modulo di elaborazione dei numeri che consente di avere nei sorgenti dei valori numerici del tipo `30e3` che le macro di scrittura trasformano in 30×10^3. Questo permette di trarre i valori numerici da file scritti dagli stessi strumenti di misura moderni, dove i numeri sono espressi con la notazione informatica dei numeri a virgola mobile.

Si può specificare il numero di cifre da scrivere nei valori numerici delle misure: il modulo di elaborazione dei numeri provvede ad arrotondare quel valore numerico al numero richiesto di decimali. Infatti se si specifica che si vogliono consistentemente 4 decimali e la virgola decimale, il numero

Capitolo 3

Implementazione su codici agli elementi finiti

Everything should be made as simple as possible, but not simpler.

Albert Einstein (1879–1955)
German physicist

In questo capitolo si utilizza la strategia di ridurre le condizioni di carico generiche agenti sul giunto ad una combinazione lineare di soluzioni note a priori (risultato del cap. 2), al fine di sviluppare un algoritmo che superi

Figura 10: Esempio di epigrafe.

`1.234567`, con il comando

```
nel preambolo
\sisetup{
  round-mode      = places,
  round-precision = 4
}

nel testo
\num{1.234567}
```

viene stampato nella forma

1,2346

dove il numero di decimali è quello voluto ma l'ultima cifra tiene conto dell'arrotondamento in alto, visto che la parte scartata è maggiore della metà dell'unità corrispondente all'ultima cifra scritta; inoltre il punto decimale è consistentemente cambiato nella virgola, come richiesto.

Il seguente frammento di codice:

```
\SI{23.4}{kg.m.s^{-2}} \\
$r=\SI{0,8768(11)e-15}{m}$ \\
\si{\joule\per\mole\per\kelvin}\\
\si{J.mol^{-1}.K^{-1}}\\
\SI{100}{\celsius} \\
\ang{1;2;3}
```

produce la di scrittura di grandezze fisiche nella forma:

$23{,}4\,\mathrm{kg\,m\,s^{-2}}$
$r = 0{,}8768(11)\cdot 10^{-15}\,\mathrm{m}$
$\mathrm{J\,mol^{-1}\,K^{-1}}$
$\mathrm{J\,mol^{-1}\,K^{-1}}$
$100\,^{\circ}\mathrm{C}$
$1^{\circ}2'3''$

9.4.4 Altri pacchetti

Per evidenziare gli ambienti matematici può essere utilizzato il pacchetto empheq. Il seguente risultato:

$$\boxed{\begin{aligned} f(x) &= ax + b && (1)\\ E &= mc^2 + \int_0^T f(t)\ \mathrm{d}t && (2)\end{aligned}}$$

è prodotto con il codice:

```
nel preambolo
\usepackage{empheq}
\newcommand*{\diff}{\mathop{}\!\mathrm{d}}

nel testo
\begin{empheq}[box=\fbox]{align}
f(x) & = a x + b \\
E & = mc^2 + \int_0^T f(t)\, \diff{t}
\end{empheq}
```

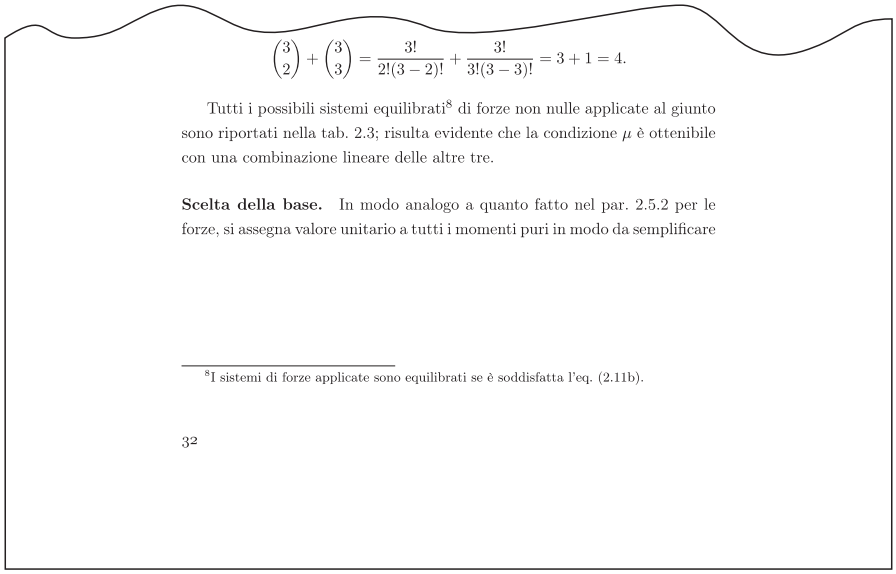
Si noti la definizione del comando `\diff` per il simbolo di differenziale: in ambiente matematico `\diff{t}` permette di ottenere 'dt' come richiesto dalle norme ISO 80000-2:2009 (2009).

Per la personalizzazione degli ambienti "tipo teorema" è necessario il pacchetto ntheorem. Il pacchetto xfrac permette invece di scrivere correttamente le frazioni nel testo e nel testo matematico (ad esempio: $5/7$).

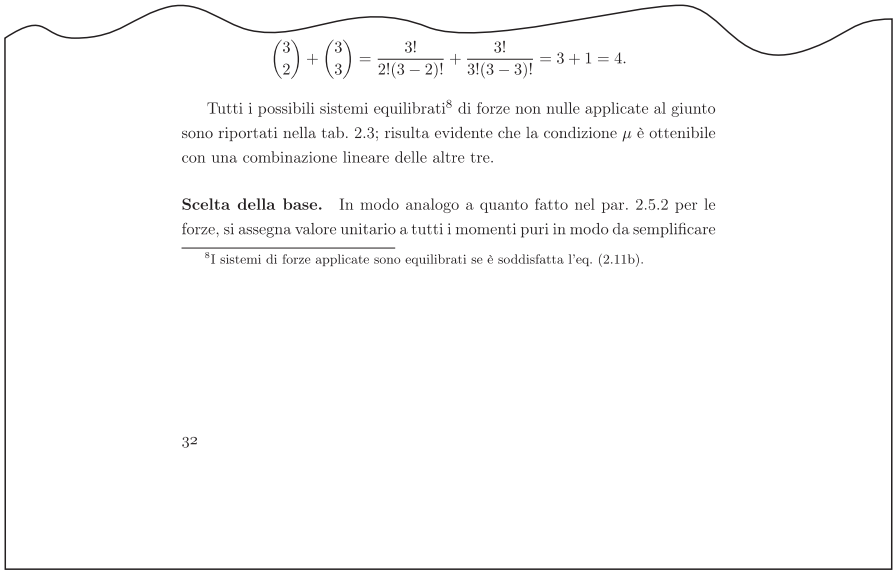
9.5 Codici ed algoritmi

Il pacchetto listings è un potente strumento con il quale si gestisce la scrittura di codici in numerosi linguaggi di programmazione, controllandone molto finemente il formato.

Per la formattazione di algoritmi sono invece consigliabili i pacchetti algorithm e algpseudocode:



(a) con opzione `bottom`



(b) senza opzione `bottom`

Figura 11: Posizione delle note.

il primo genera degli oggetti flottanti mentre il secondo no.

9.6 Riferimenti incrociati

In molti casi è comodo usare contemporaneamente i comandi `\ref` e `\pageref` per riferirsi a figure e tabelle, specialmente quando ci sono più pagine tra il riferimento e l'oggetto. Per questo, alcuni utenti utilizzano comandi come

```
\newcommand{\fullref}[1]{%
    \ref{#1} a pagina~\pageref{#1}}
```

che semplifica la scrittura del riferimento. Tuttavia, non sapendo a priori dove sia posizionato l'oggetto a cui ci si riferisce, utilizzando un comando del genere può capitare che il `\pageref` punti alla pagina stessa dove si trova il riferimento producendo un risultato insoddisfacente.

Per rendere automatica la scrittura dei riferimenti completi è possibile utilizzare il pacchetto varioref che introduce il comando `\vref` da usarsi nello stesso modo del comune `\ref`. Tale pacchetto funziona in parallelo a babel e quindi si adatta alla lingua utilizzata nel testo. Ad esempio

```
si veda la figura~\vref{fig:Mia:Figura}
```

produce, a seconda di dove viene posizionata la figura, qualcosa del tipo

si veda la figura 3.1 nella pagina successiva

oppure

si veda la figura 3.1 a pagina 24

Per quanto riguarda invece il riferimento ad equazioni, è consigliabile utilizzare il comando `\eqref{...}` di amsmath al posto di `(\ref{...})`. Ad esempio

```
... grazie all'equazione~\eqref{e2}
```

produce qualcosa del tipo

... grazie all'equazione (3.6)

Un pacchetto alternativo e per certi aspetti più potente di varioref è il pacchetto cleveref. Si rimanda il lettore al manuale d'uso per approfondimenti.

9.7 Revisione del codice

In fase di revisione del codice è molto utile, oltre ad un'attenta lettura del file di *log* dei messaggi (`.log`), l'utilizzo dei pacchetti refcheck e showkeys che controllano l'utilizzo dei `\label` e dei `\ref`. In aggiunta a questi è anche conveniente abilitare l'opzione `draft` per la `documentclass`: in questo modo i punti in cui il testo sborda dai margini verranno evidenziati con delle barre nere.

10 Siti utili

In aggiunta alle guide ed ai manuali citati nella bibliografia, sono disponibili sul Web una serie di risorse utili per risolvere i problemi incontrati durante l'utilizzo di LaTeX.

Il riferimento primario per la comunità italiana di utenti LaTeX è il sito del Gruppo Utilizzatori Italiani di TeX (GuIt)[28] che ospita un forum sull'argomento.[29] ed ha anche una sezione documentazione ben organizzata.

Altro sito di interesse è quello del CTAN che ospita gran parte del materiale su LaTeX disponibile in rete ed è dotato di un motore di ricerca.

Sarovar[30] è un catalogo molto completo di pacchetti e programmi legati a TeX e LaTeX. Permette svariati tipi di ricerca, in particolare è estremamente utile la lista "topical" quando non si conosce il nome di un pacchetto ma solo "quello che deve fare".

Altri due importanti riferimenti sono il sito di domande e risposte su StackExchange[31] e il sito LaTeX Community.[32]

Riferimenti bibliografici

Beccari, C. (2012). *Introduzione alla definizione della geometria della pagina.* `http://www.guitex.org/home/images/doc/GuideGuIT/intropagedesign.pdf`.

— (2013). *La classe `TOPtesi`. Per comporre la tesi al Poli e in molte altre università.* `http://texdoc.net/texmf-dist/doc/latex/toptesi/toptesi-doc-xetex.pdf`.

Beccari, C. e Gordini, T. (2012). *Codifiche in TeX e LaTeX. Dal sorgente al PDF, guida pratica per lavorare con successo.* `http://www.guitex.org/home/images/doc/GuideGuIT/introcodifiche.pdf`.

Beccari, C., Canavero, F., Rossetti, U. e Valabrega, P. (2011). *Saper Comunicare. Cenni di scrittura tecnico-scientifica.* Politecnico di Torino, Torino, Italy. URL `https://didattica.polito.it/tesi/SaperComunicare.pdf`. Ver. 1.11.

Biccari, F. (2012). *Documentation of the LaTeX class `sapthesis.cls`.* `http://texdoc.net/texmf-dist/doc/latex/sapthesis/sapthesis-doc.pdf`.

Cevolani, G. (2006). «Norme tipografiche per l'italiano in LaTeX». *ArsTEXnica*, **1** (1).

Eco, U. (1977). *Come si fa una tesi di laurea.* Bompiani, Milano.

Gregorio, E. (2011). *Introduzione a XeLaTeX.* `http://profs.sci.univr.it/~gregorio/introxelatex.pdf`.

— (2013). *Il pacchetto `frontespizio`.* `http://texdoc.net/texmf-dist/doc/latex/frontespizio/frontespizio.pdf`.

Gruppo Utilizzatori Italiani di TeX (2013). *Introduzione all'arte della composizione tipografica con LaTeX.* `http://www.guitex.org/home/images/doc/GuidaGuIT-B5.pdf`.

ISO 80000-2:2009 (2009). *Quantities and units — Part 2: Mathematical signs and symbols to be used in the natural sciences and technology.* International Organization for Standardization.

Kohm, M. e Morawski, J.-U. (2012). *The `KOMA-Script` guide.* `http://texdoc.net/texmf-dist/doc/latex/koma-script/scrguien.pdf`.

Lehman, P. (2013). *The `biblatex` package.* `http://texdoc.net/texmf-dist/doc/latex/biblatex/biblatex.pdf`.

Lesina, R. (2013). *Il nuovo manuale di stile. Edizione 2.0. Guida alla redazione di documenti, relazioni, articoli, manuali, tesi di laurea.* Zanichelli, Bologna. Ristampa della II edizione 1994.

Matricciani, E. (2000). *La tesi scientifica. Guida alla comunicazione in Ingegneria e nelle Scienze.* Paravia Scriptorium, Torino.

— (2003). *Fondamenti di comunicazione tecnico-scientifica.* Apogeo, Milano.

28. `http://www.guitex.org`
29. `http://www.guitex.org/home/it/forum/index`
30. `http://texcatalogue.sarovar.org`
31. `http://tex.stackexchange.com`
32. `http://www.latex-community.org`

— (2007). *La comunicazione tecnico-scientifica*. Aracne editrice, Milano.

Miede, A. (2013). *A classic thesis style*. http://texdoc.net/texmf-dist/doc/latex/classicthesis/ClassicThesis.pdf.

Mori, L. (2007). «Scrivere la tesi di laurea con LaTeX 2ε». *ArsTEXnica*, **1** (3).

Mori, L. F. (2006). «Tabelle su LaTeX 2ε: pacchetti e metodi da utilizzare». *ArsTEXnica*, (2), pp. 31–47.

Pakin, S. *The comprehensive LaTeX symbol list*. In http://texdoc.net/texmf-dist/doc/latex/comprehensive/symbols-a4.pdf.

Pantieri, L. e Gordini, T. (2011). *L'arte di scrivere con LaTeX*. URL http://www.lorenzopantieri.net/LaTeX_files/ArteLaTeX.pdf.

Umeki, H. (2010). *The geometry package*. http://texdoc.net/texmf-dist/doc/latex/geometry/geometry.pdf.

UNI-ISO 5966 (1989). *Presentazione dei rapporti scientifici e tecnici*. Ente Italiano di Unificazione, Milano.

Valbusa, I. (2013). *User's guide to suftesi. A document class for typesetting theses, books and articles*. http://texdoc.net/texmf-dist/doc/latex/suftesi/suftesi.pdf.

Voss, H. (2010). «Math mode». Mathmode.pdf in http://texdoc.net/texmf-dist/doc/latex/mathmode/Mathmode.pdf.

Voss, H. (2013). *Package hvfloat. Rotating objects and captions*. http://texdoc.net/texmf-dist/doc/latex/hvfloat/hvfloat.pdf.

Wilson, P. (2010). *The memoir class for configurable typesetting*. http://texdoc.net/texmf-dist/doc/latex/memoir/memman.pdf.

▷ Agostino De Marco
Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Industriale
agostino dot demarco at unina dot it

LaTeX e le lingue romanze alpine: il piemontese

Claudio Beccari

Sommario

LaTeX può gestire molte lingue, attualmente quasi un'ottantina. Alcune delle lingue gestibili da LaTeX sono usate da poche persone, altre da milioni. Questo articolo vorrebbe essere un altro passo verso l'estensione di LaTeX alle lingue romanze usate nelle regioni alpine d'Europa. In questo articolo sull'argomento si parla del piemontese, parlato, letto e scritto da alcuni milioni di persone nel Nord Ovest d'Italia e nelle comunità piemontesi sparse per il mondo.

Abstract

LaTeX is used to typeset in a variety of languages: at the time of writing, it can handle some 80 languages, some of them used by very few people, some by millions. At the moment, there are few facilities for typesetting documents in the various more or less official Alpine Romance languages. This paper would be a step in order to extend the LaTeX language capabilities to the various romance languages that are being used by large communities in the European Alps. In this article on the subject we deal with Piedmontese, spoken, read, an written by some million people in North-western Italy and within the Piedmontese communities around the world.

Somari

LaTeX a peul gestì un baron 'd lengue, atualment d'apopré otanta. Quaichidun-a 'd coste lengue a l'é parlà da pòche përson-e, àotre da milion. Cost artìcol-sì veul esse n'àotr pass anans vers l'estenssion dë LaTeX a le lengue romanse dovrà ant le region alpin-e d'Europa. Ant cost artìcol-sì as parla dël piemontèis, parlà, let e scrit da quaich million ëd përson-e ant el nòrd-òvest d'Italia e ant le comunità piemontèise spatarà ant ël mond.

1 Introduzione

In precedenti articoli, (Beccari, 2012) e (Beccari, 2013), ho parlato dell'estensione di LaTeX alla gestione del romancio svizzero. e a quella del friulano Qui ora mi occuperò del piemontese.

Ci sono due motivi importanti: lo devo alla mia regione di adozione dove vivo da circa 60 anni; lo devo ad una lingua che nella sua forma più arcaica, documentata da scritti importanti, è forse anteriore all'italiano, o al toscano come lo si chiamava allora.

Nel 1150 circa furono pubblicati i *Sermon supalpengh* (Delfuoco e Bernardi, 2004); si trattava di 22 sermoni completi a commento della liturgia, scritti come testo didattico per le scuole che formavano i templari. Non bisogna scordasi che la Via Francigena attraversava il Piemonte e che lungo questo percorso di pellegrinaggio esistevano un gran numero di monasteri ed abbazie che sostenevano i pellegrini che la percorrevano generalmente a piedi per scendere dalle terre dei Franchi fino a Roma. Molte di queste strutture erano gestite dai templari, i quali dovevano rivolgersi non solo ai pellegrini d'oltralpe ma anche ai residenti appartenenti al *vulgus*, che parlavano non più il latino, ma una lingua gallo-italica, che oggi chiamiamo piemontese.

La scrittura era quella "arcaica", ma rendeva abbastanza bene i suoni di questa lingua.

Dante, nel suo *De vulgari eloquentia* (Alighieri, 1990), esamina l'origine dei vari "volgari" alla ricerca del più nobile e degno di essere usato da tutte le persone colte. Per le lingue di Trento, Torino e Alessandria dice testualmente:

> Quare, cribellum cupientes deponere, ut residentiam cito visamus, dicimus, Tridentum atque Taurinum, nec non Alexandriam civitates, metis Ytalie in tantum sedere propinquas, quod puras nequeunt habere loquelas, ita quod, si etiam quod turpissimum habent vulgare haberent pulcerrimum, propter aliorum commistionem esse vere latinum negaremus; quare, si latinum illustre venamur, quod venamur in illis inveniri non potest.

che tradotto in "volgare" suona come[1]:

> Per cui, desiderosi come siamo di deporre il setaccio, e per dare uno sguardo veloce alla rimanenza, diciamo che le città di Trento e di Torino, nonché di Alessandria, sono situate talmente vicino ai confini d'Italia che non possono avere parlate pure; tanto che, se anche possedessero un bellissimo volgare — e invece l'hanno bruttissimo —, per come è mescolato coi volgari di altri popoli dovremmo negare che si tratti di una lingua veramente italiana. Perciò, se quello che cerchiamo è

1. Ho ricavato i due testi latino e italiano dal sito `http://www/danteonline.it`; la traduzione non è mia e non è certo una traduzione letterale; il concetto espresso però è quello del testo latino.

> 1'italiano illustre, 1'oggetto della nostra ricerca non si può trovare in quelle città.

Dante, che capiva il catalano e l'occitano (il cui nome fu coniato da lui stesso e ancora oggi le comunità occitane lo ricordano per questo riconoscimento), evidentemente o non capiva il piemontese, o non lo considerava abbastanza vicino al latino da considerarlo una lingua nobile. Strano che giudicasse il torinese come tanto vicino a volgari di altri popoli (che parlavano l'occitano, che Dante conosceva ed apprezzava) da essere mescolato al punto tale da non potersi considerare una lingua veramente nobile.

Tuttavia la lingua piemontese, checché ne dica Dante, appare sicuramente strana a chi non la capisce – come succede con qualunque lingua che non si conosca –, ma se appena appena la si capisce è una lingua che, almeno alle mie orecchie, suona dolcissima, specialmente se parlata da signore di una certa età sedute al tavolino di qualche caffè storico del centro mentre si esprimono in *torinese cortigiano*.

Cos'è questo strano *torinese cortigiano*? È la varietà di piemontese che si parlava alla corte dei Savoia e che è servita a Maurizio Pipino, medico e filologo, per redigere la prima grammatica piemontese nel 1783 (Pipino, 1783), dedicandola a Sua Altezza reale Maria Adelaide Clotilde Saveria di Francia Principessa del Piemonte, sposa di Carlo Emanuele IV di Savoia. Questa principessa, diventata Regina di Sardegna, dopo la restaurazione post napoleonica, arrivata nella corte di Torino, volle imparare il piemontese e lo usò, sembra, con molta padronanza e con continuità a corte. Chiaramente a corte si parlava un piemontese raffinato, ben diverso dal piemontese parlato in provincia e nelle campagne, come rileva il Pipino il quale definì cortigiana questa varietà raffinata.

Pipino si preoccupò di standardizzare l'ortografia introducendo un certo numero di regole e di segni diacritici per differenziare certi suoni; fu dunque sua la prima grammatica "ufficiale" del piemontese. Questo avveniva circa trecento anni dopo che era stata scritta la prima grammatica del toscano, ormai diventato l'italiano conosciuto e condiviso dalle persone colte. Sono rimasto stupito che questa prima grammatica italiana fosse stata scritta nella seconda metà del secolo XV da Leon Battista Alberti, che è molto conosciuto come architetto eccelso; egli fu anche un genio poliedrico che si occupò di moltissime cose; non solo scrisse la *Grammatica della lingua toscana* (Alberti, 1973), ma fu anche il padre della crittografia moderna; purtroppo scrisse il suo lavoro *De componendis cyfris* (Alberti, 1998) come strumento per cifrare i documenti delle cancellerie pontificie; a causa di ciò, questo testo è rimasto chiuso nel segreto degli archivi pontifici per moltissimo tempo, tanto che quando fu ritrovato, i matematici di due o tre secoli dopo avevano riscoperto autonomamente la stessa teoria di cifratura.

La Gramatica (sic!) del Pipino servì moltissimo per lo sviluppo della lirica, del romanzo, della tragedia e dell'epica; il piemontese era già usato (sia pure in una grafia più antica) nella letteratura religiosa, specialmente nelle Comunità Valdesi, dove la lettura del Vangelo, (Chiesa Valdese, 1834), è affidata al singolo credente e perciò deve essere scritto nella lingua che il credente usa parlando. Benché la Chiesa Valdese abbia il suo centro culturale e spirituale nella Val Pellice, il Vangelo citato fu stampato a Londra perché nel Regno di Sardegna non era ancora stato emanato lo Statuto Albertino (1848) che sanciva la libertà religiosa[2]. Benché scritto dopo la pubblicazione della grammatica di Maurizio Pipino, l'ortografia di questo Vangelo rispecchia quella di scritti precedenti, dove il suono /u/ viene reso col dittongo alla francese 'ou', e 'o' conserva il suo suono /o/.

Oggi il piemontese ha un codice nella norma ISO 639-3 e la sigla di tre lettere pms (ISO, 2007): è riconosciuto fra le lingue minoritarie europee fin dal 1981 (Rapporto 4745 del Consiglio d'Europa) ed è inoltre censito dall'UNESCO (Red book on endangered languages) tra le lingue meritevoli di tutela. È classificata come lingua e non come dialetto, essendo un idioma che non si può considerare una variante dell'italiano, e appartiene al gruppo delle lingue gallo-italiche, considerate da alcuni studiosi lingue di transizione rispetto alle lingue gallo-romanze.

Bisogna però dire che il piemontese è nato su un substrato celtico, con adstrati occitani e longobardi, e con superstrati vari dovuti alle varie occupazioni del territorio; ma importantissimo è il superstrato italiano, perché questa lingua è diventata ufficiale in Savoia e poi nel Regno di Sardegna già dal XVI secolo per ogni scritto di carattere legale e amministrativo, oltre che culturale; l'Università di Torino nel 1566 fu riformata sul modello dell'Università di Bologna e adottò sin da allora l'italiano per l'amministrazione, accanto al latino del mondo accademico. La penetrazione dell'italiano in Piemonte, quindi, dura da molto tempo; la situazione è "precipitata" negli anni '50 e '60, quando la popolazione di Torino quasi è raddoppiata grazie all'immigrazione di un numero impressionante di persone giunte da altre regioni, specialmente dalle regioni dell'Italia nord-orientale, centrale e meridionale. È per questo che l'UNESCO la considera fra le lingue degne di particolare tutela; esistono previsioni che questa lingua possa estinguersi nel

2. Sono andato a cercare il testo completo dello Statuto Alberino; la libertà religiosa era molto limitata, nel senso che ogni religione diversa dalla Cattolica Romana era *tollerata* (articolo 1); per quel che riguarda la stampa di libri di argomento religioso continuava invece ad essere richiesto il preventivo permesso del Vescovo (articolo 28); è facile capire perché le comunità ebree e protestanti si rivolgessero spesso all'estero.

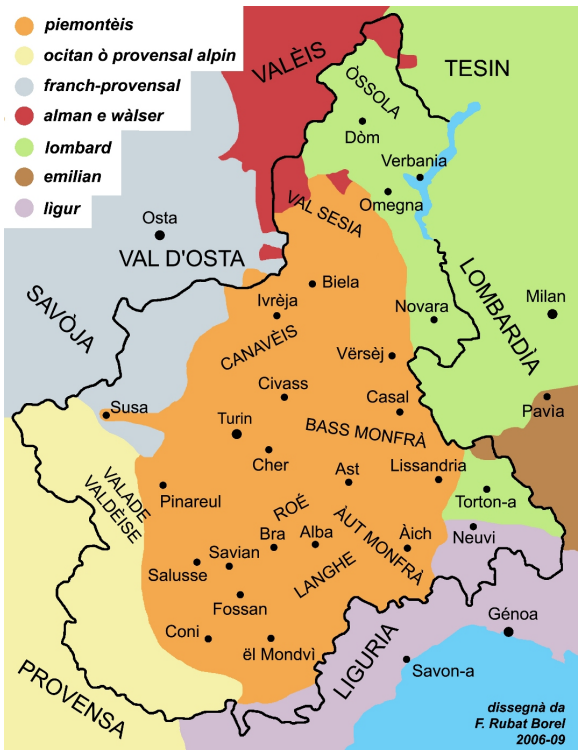


FIGURA 1: Carta geolinguistica del Piemonte e delle regioni circostanti

corso del XXI secolo. Mi auguro vivamente che questo non succeda.

Nello stesso tempo va notato che il piemontese, con i suoi dialetti, non è l'idioma parlato in tutta la regione; la carta geolinguistica disegnata da F. Rubat-Borel tra il 2006 e il 2009, figura 1, mostra chiaramente la suddivisione territoriale delle varie lingue che vengono usate in Piemonte.

Si noti che nella zona marcata con la dicitura *Valade Valdèise* la religione valdese non è l'unica religione praticata, né corrisponde ad un territorio linguisticamente omogeneo; vi si parlano, oltre all'italiano come lingua ufficiale, il piemontese, l'occitano e il francese. La zona marcata con la dicitura *Valèis* corrisponde solo in parte alla zona germanofona del cantone Vallese/Valais/Wallis della vicina Svizzera, e comprende alcune zone del Piemonte e della Valle d'Aosta dove si parla il walser (lingua alto allemannica). Il ligure, l'emiliano e, specialmente, il lombardo occidentale occupano buona parte delle zone meridionali e orientali della regione Piemonte. Il franco-provenzale svizzero, savoiardo e valdostano si espande nel Piemonte nord-occidentale.

Con queste varietà di idiomi, la Regione Piemonte ha emanato la legge regionale 11 del 2009 nella quale sono predisposte le "Norme per la tutela la valorizzazione e la promozione del patrimonio linguistico del Piemonte" e l'elenco di queste lingue è: l'occitano, il franco-provenzale, il francese e il piemontese. La sentenza della Corte Costituzionale 170 del 2010 dichiara incostituzionale l'inclusione del piemontese fra le lingue meritevoli di essere tutelate e valorizzate, in quanto la Costituzione affida ad apposita legge la definizione delle lingue storiche da tutelare, cioè alla legge 482 del 1999, che afferma: «In attuazione dell'articolo 6 della Costituzione e in armonia con i principi generali stabiliti dagli organismi europei e internazionali, la Repubblica tutela la lingua e la cultura delle popolazioni albanesi, catalane, germaniche, greche, slovene e croate e di quelle parlanti il francese, il franco-provenzale, il friulano, il ladino, l'occitano e il sardo». Al parlamento un certo numero di deputati ha presentato la proposta di legge C. 4181 nel marzo 2011 per modificare la legge 482/1999 al fine di aggiungere alle lingue riconosciute dallo Stato come meritevoli di tutela e valorizzazione anche il piemontese; è molto interessante leggere il preambolo a questa proposta di legge nel documento CAVALLOTTO *et al.* (2011).

È chiaro che c'è del contenzioso nel considerare il piemontese una lingua degna di tutela, piuttosto che un dialetto dell'italiano: l'eventuale estinzione del piemontese, in mancanza di tutela, non interesserebbe la Repubblica. Non entro in queste questioni, anche se mi rammarico che la situazione sia arrivata a questo punto. Per mio conto io sono interessato a rendere LaTeX adatto per comporre in piemontese e mi auguro che questo fatto venga recepito anche dai molti che ancora oggi scrivono in questa lingua, per metterli in condizione di usare uno strumento moderno e utile come il sistema TeX.

2 Ortografia del piemontese

Tornando al piemontese, l'ortografia della lingua fu ammodernata e semplificata negli anni '20 del XX secolo dalla Companìa dij Brandé fondata da Pinin Pacòt, e poi codificata nella grammatica di Camillo Brero (BRERO, 2008a), che produsse anche un vocabolario aggiornato ai tempi moderni (BRERO, 2008b).

Questa modernizzazione consiste in una certa semplificazione dell'ortografia di Pipino, ma sostanzialmente sono stati ridotti un poco i segni diacritici, specialmente quelli che a stampa (o con le vecchie macchine da scrivere) sono o erano difficili da marcare.

Delle 26 lettere dell'alfabeto latino se ne usano 22, come per l'italiano "puro" ma con l'aggiunta della lettera 'j'; due segni totalmente diversi dall'italiano sono 'n-' e 's-', il secondo dei quali, in verità, potrebbe sembrare superfluo visto che serve per scrivere e pronunciare staccato il gruppo consonantico 'sc' seguito da 'e' o 'i', come in 's-centrà', visto che in piemontese puro è assente il suono dell'italiano "scena"; data, però, la frequenza degli italianismi, che conservano la grafia italiana, l'uso particolare di 's-' non è più superfluo. Per una

Tabella 1: Corrispondenza fra fonemi e grafemi

Fonemi IPA	Grafemi	Fonemi IPA	Grafemi
a	a, à	wa, we, wi, wo	ua, ue, ui, uo
e, ɛ, æ	e	k	c, ch
ɛ, æ	è	tʃ	c, ci, cc
e	é	kw	qu
g	ë	g	g, gh
i	i, ì	dʒ	g, gi, gg
u	o, ó	ɲ	gn
ɔ	ò	s	s, ss, s-
y	u, ù	z	s
aɪ	ai	b	b
aje	aje	d	d
eu	ø	f	f
aju	ajo	l	l
uje,	oje	m	m
ja	ja, ia	n, ŋ	n
je	je, ie	ŋ	n-
jo	jo, io	p	p
ju	ju, iu	qw	qu
aʊ, iʊ, øʊ	au, iu, euv	r	r
aj, ej, ij, oj, uj, øj	ai, ej, ij, oj, uj, euj	v, ʊ	v, V

visone globale dei fonemi e dei grafemi piemontesi ci si può riferire alla tabella 1.

Rispetto all'italiano compaiono due fonemi in più fra le vocali: /y/ e /ø/; e manca il fonema /o/, che ricompare negli italianismi, ma è assente dal piemontese puro. Così fra i fonemi consonantici compare il fonema /ŋ/ ma mancano i fonemi /ʃ/, /ʎ/, /ts/ e /dz/, che però riappaiono negli italianismi; secondo alcuni il fonema /ɛ/ è ancora più aperto (/æ/) che in italiano.

Dal punto di vista dell'ortografia si noti che si usa il trattino dopo la 's' e dopo la 'n' per indicare i suoni particolari indicati nella tabella 1, e questo pone un problema in LaTeX perché il trattino svolge diverse funzioni; in particolare, come trattino copulativo, cioè di unione fra due parole, impedisce la sillabazione fra le due parole, consentendo di andare a capo solo dopo il trattino. In piemontese si potrebbe usare il trattino anche dopo 'cc' e 'gg' quando compaiono all'interno di una parola per indicare il suono palatale indicato nella tabella 1; il problema con LaTeX persiste anche in questo caso.

La lettera 'j' sostituisce generalmente quello che in italiano si rappresenterebbe con il trigrafo 'gli'; ma esistono dei casi in cui è necessario distinguere bene nel rendere il fonema /aɪ/ o /aj/: per esempio: 'ai', 'a-i' 'a-j', 'aj', rispettivamente: (*a*) 'ai' corrisponde alla preposizione articolata italiana 'ai', (*b*) 'a-i' corrisponde al soggetto verbale (inesistente in italiano) seguito dall'avverbio di luogo, in italiano 'ci' oppure 'vi', in frasi come 'ci sono due bimbi': 'a-i son doi cit'; (*c*) il soggetto verbale seguito dal pronome proclitico 'chila a-j veul bin' (lei gli vuole bene); (*d*) 'aj' è il sostantivo corrispondente all'italiano 'aglio'. Questi sono problemi di grafia delicati, che però non hanno nessuna influenza sul comportamento di LaTeX in presenza del trattino.

Si noti che il suono gutturale e palatale delle lettere 'c' e 'g' viene gestito come in italiano; cioè esse hanno il suono palatale prima di 'e' ed 'i' e il suono gutturale prima di 'a', ò', 'o'; nel primo caso il suono diventa gutturale interponendo 'h' e nel secondo caso il suono diventa palatale interponendo 'i'. Siccome queste lettere possono apparire anche alla fine di una parola, le si rende gutturali aggiungendo una 'h' alla fine, oppure le si rende palatali raddoppiando 'c' o 'g'. Dal punto di vista di LaTeX questo non è assolutamente un problema. Per altro queste lettere raddoppiate non compaiono altrove nelle parole del piemontese puro e nemmeno negli italianismi.

Invece la 's' che si pronuncia sorda quando è adiacente a un'altra consonante, mentre si pronuncia sonora quando è intervocalica, viene sostituita dalla 'z' quando deve essere pronunciata sonora nelle posizioni in cui la regola la darebbe sorda, per esempio 'zenziva' (gengiva); al contrario la 's' viene raddoppiata in 'ss' (sorda ma non pronunciata doppia) quando la regola la darebbe sonora, per esempio 'stassion' (stazione). Se ne deduce che il digrafo 'ss' indica un solo fonema non raddoppiato e forse non andrebbe diviso fra due sillabe, come avviene invece in italiano. Però la grammatica del Brero, come anche la grammatica del Pipino, danno per scontato che si conosca la divisione in sillabe, ma non indicano mai quali siano le regole per questa divisione. Tutto lascia pensare che siano

sostanzialmente le stesse che si usano in italiano, solo che il piemontese presenta gruppi consonantici che in italiano non esistono, quindi chi vuole preparare i pattern di sillabazione per LaTeX si trova in imbarazzo. Un esempio è proprio il digrafo 'ss' indicato sopra; in italiano non esistono digrafi contenenti due lettere uguali; le doppie italiane non sono propriamente dei digrafi, ma rappresentano delle consonanti rafforzate, e in italiano le doppie si dividono fra due sillabe adiacenti. In piemontese sembra che le cose siano diverse. In friulano (Beccari, 2013) il digrafo 'ss' non è divisibile fra due sillabe; non dico che in piemontese ci si debba comportare nello stesso modo; lo indico solo per segnalare che di fronte a questo problema si possono prendere decisioni opposte.

Inoltre in piemontese ha luogo assai spesso la *prostesi*; in italiano si usava scrivere una volta la 'i' prostetica all'inizio di parole che iniziavano con la 's impura' ed erano precedute da una preposizione terminante in consonante, per esempio 'in iscuola', 'per istrada'. Oggi in italiano forse si pronuncia ancora quella 'i', ma si raccomanda di non scriverla. In piemontese la vocale prostetica è 'ë' e si usa abbastanza spesso, per esempio 'l'ëstudent'; ma l'uso è talmente radicato che si perde la nozione della prostesi e si scrive anche 'lë student'; queste varianti ortografiche, però, non interferiscono con LaTeX.

La grafia del piemontese non pone altri problemi particolari, oltre a quelli già descritti.

3 LaTeX e il piemontese

LaTeX, come anche gli altri mark-up del sistema TeX, da XeLaTeX, a ConTeXt a LuaTeX, gestisce le lingue in due modi distinti.

Da un lato deve essere in grado di dividere in sillabe le parole in fin di riga per eseguire la giustificazione del testo.

Dall'altro deve realizzare regole tipografiche particolari per ciascuna lingua (come le spaziature indivisibili prima dei segni di interpunzione 'alti' quando si compone in francese) e produrre le parole "infisse" come 'Capitolo', 'Chapter' 'Chapître', 'Capìtol'; assieme a queste diversificazioni linguistiche deve saper produrre la data con i nomi dei mesi "infissi" ma scritti con la grafia corretta (iniziale maiuscola o minuscola; varianti).

Per fare questo sono necessari diversi file; per il primo compito occorre il file dei pattern di sillabazione (che è accompagnato da un paio di altri file necessari per gestire le particolarità dei caratteri usati nella lingua) e per il secondo compito occorre il file di descrizione della lingua.

4 File di descrizione della lingua e file di pattern

La funzione descrittiva della lingua esposta nella sezione precedente è svolta da (generalmente un solo) file con estensione `.ldf`. Invece i pattern possono richiedere diversi file: diversi perché devono essere usati solo con font aventi codifiche del tipo T1, oppure con codifiche UNICODE; c'è modo di costruire file che soddisfino entrambe le esigenze, ma la predisposizione dei pattern è piuttosto delicata.

I file di descrizione della lingua sono gestiti dai pacchetti babel (Braams, 2011) o da polyglossia (Charette, 2011); i file di pattern sono usati solo nella generazione dei file di formato, che hanno estensione `.fmt`, dei quali la maggior parte degli utenti del sistema TeX ignora l'esistenza. Una installazione del sistema TeX ben fatta fa sì che in effetti l'utente non abbia nessun bisogno di conoscerne l'esistenza; però questa mancanza di conoscenza, talvolta, può dare luogo a errori strani e a ricerche di aggiustamenti che inducono solo frustrazione, in quanto la sillabazione prodotta dai pattern è gestita solo dai file di formato e babel e polyglossia si occupano solamente di selezionare i pattern già presenti nei file di formato.

4.1 Il file di descrizione della lingua

I file `.ldf` per babel hanno il nome formato con quello della lingua e con l'estensione indicata. I file per polyglossia hanno il nome formato agglutinando il nome della lingua con il prefisso `gloss-`; il nome della lingua è solitamente il nome in inglese e in lettere minuscole; per il piemontese si hanno perciò i file `piedmontese.ldf` e `gloss-piedmontese.ldf`.

La funzione del file `.ldf` nei due casi è sostanzialmente la stessa, ma mentre con babel la lingua è una opzione per il pacchetto, con polyglossia sono disponibili alcuni comandi con i quali l'utente può precisare le singole lingue da usare nel documento specificandone mediante opzioni anche alcune particolarità che sono disponibili grazie all'uso dei font OpenType. Non si scenderà in questi dettagli; si descriverà il file `piedmontese.ldf` che contiene le definizioni importanti e assolutamente necessarie in ogni file di questo genere.

```
\LdfInit\CurrentOption{captions\CurrentOption}
\ifx\l@piedmontese\@undefined
    \@nopatterns{piedmontese}
    \ifx\l@italian\@undefined
        \adddialect\l@piedmontese\l@english
    \else
        \adddialect\l@piedmontese\l@italian
    \fi
\fi
\@namedef{captions\CurrentOption}{%
    \def\prefacename{Prefassion}%
    \def\refname{Riferiment}%
    \def\abstractname{Somari}%
    \def\bibname{Bibliograf\‘ia}%
    \def\chaptername{Cap\‘itol}%
```

```
16      \def\appendixname{Gionta}%
17      \def\contentsname{T\‘aula}%
18      \def\listfigurename{Lista dle figure}%
19      \def\listtablename{Lista dle tabele}%
20      \def\indexname{T\‘aula anal\‘itica}%
21      \def\figurename{Figura}%
22      \def\tablename{Tabela}%
23      \def\partname{Part}%
24      \def\enclname{Gionta/e}%
25      \def\ccname{Con c\‘opia a}%
26      \def\headtoname{P\"er}%
27      \def\pagename{P\‘agina}%
28      \def\seename{v\"ed}%
29      \def\alsoname{v\"ed anche}%
30      \def\proofname{Dimostrassion}%
31      \def\glossaryname{Glossari}%
32  }
33  \@namedef{date\CurrentOption}{%
34  \def\today{\number\day\space\ifcase\month\or
35  \"ed gen\’e\or \"ed fevr\’e\or \"ed mars\or d’
        avril\or
36  \"ed maj\or \"ed giugn\or \"ed luj\or d’agost\or
37  d\"e st\‘ember\or d’ot\’ober\or \"ed nov\‘ember\or
38  d\"e dz\‘ember\fi\space dal\space\number\year}}
39
40  \providehyphenmins{\CurrentOption}{\tw@\tw@}
41
42  \expandafter\addto\csname extras\CurrentOption
        \endcsname{%
43    \babel@savevariable\clubpenalty
44    \babel@savevariable\widowpenalty
45    \babel@savevariable\@clubpenalty
46    \clubpenalty3000\widowpenalty3000\@clubpenalty
        \clubpenalty}%
47  \expandafter\addto\csname extras\CurrentOption
        \endcsname{%
48    \babel@savevariable\finalhyphendemerits
49    \finalhyphendemerits50000000}%
50  \expandafter\addto\csname extras\CurrentOption
        \endcsname{%
51    \lccode‘’=‘’}%
52  \expandafter\addto\csname noextras\CurrentOption
        \endcsname{%
53    \lccode‘’=0}%
54  %
55  \initiate@active@char{"}%
56  \expandafter\addto\csname extras\CurrentOption
        \endcsname{%
57  \bbl@activate{"}\languageshorthands{piedmontese}}%
58  \declare@shorthand{piedmontese}{"}{%
59  \ifmmode
60      \def\pms@next{’’}%
61  \else
62      \def\pms@next{\futurelet\pms@temp\pms@cwm}%
63  \fi
64  \pms@next
65  }%
66  \def\pms@@cwm{\nobreak\discretionary{-}{}{}
        \nobreak\hskip\z@skip}%
67  \DeclareRobustCommand*{\pms@cwm}{\let\pms@@next
        \relax
68  \ifcat\noexpand\pms@temp a%
69      \def\pms@@next{\pms@@cwm}%
70  \else
71      \if\noexpand\pms@temp\string/%
72          \def\pms@@next{\slash\@gobble}%
73      \else
74          \if\pms@temp-%
75              \def\pms@@next{\hskip\z@\nobreak-\hskip
        \z@\nobreak\@gobble}%
76          \else
77              \ifx\pms@temp"%
78                  \def\pms@@next{‘‘\@gobble}%
79              \fi
80          \fi
81      \fi
82  \fi
83  \pms@@next}%
84
85  \expandafter\addto\csname noextras\CurrentOption
        \endcsname{%
86  \bbl@deactivate{"}}
87
88  \ldf@finish\CurrentOption
89  \endinput
```

Alcuni commenti sono importanti. Innanzi tutto nel file la lingua “piedmontese” non viene quasi mai menzionata esplicitamente, ma è sostituita dalla macro `\CurrentOption` che, in quanto opzione specificata a babel o a polyglossia, contiene il nome della lingua scelta, in questo caso il piemontese nella sua ortografia inglese. La prima linea di codice serve per l’inizializzazione, ma fra le altre cose verifica che per “piedmontese” sia stato definito `\captionspiedmontese`; ciò serve a babel tra l’altro per controllare se il file sia già stato caricato.

Le righe da 2 a 9 verificano se esista nel file di formato un insieme di pattern di sillabazione collegato alla lingua piemontese; se un tale insieme non esistesse, sarebbe necessario comunque dare un significato al comando `\l@piedmontese` associandolo all’insieme di un’altra lingua come se ne fosse una varietà. Si è preferito associarlo all’insieme dell’italiano, perché le due lingue hanno un certo grado di affinità; tuttavia in una distribuzione incompleta del sistema TEX anche i pattern della lingua italiana potrebbero mancare dal formato, quindi, come ultima risorsa, si mette il piemontese come dialetto dell’inglese (sic!), che d’altronde è quello che si fa per ogni lingua, visto che i pattern dell’inglese sono sempre presenti nel formato.

Le successive righe dalla 10 alla 32 definiscono tutte le parole fisse che compaiono nei titolini o in certi ambienti; si sono presi questi nomi dal vocabolario bilingue piemontese–italiano di Brero (2008b). Si noti che in queste parole, come nei successivi nomi dei mesi, il file non contiene altro che caratteri ASCII a 7 bit, per cui questo file non dipende da nessuna codifica particolare di input. Nel corrispondente file `gloss-piedmontese.ldf`, invece, quelle parole sono state scritte con le lettere accentate in codifica utf8, perché quei file vengono letti solo con polyglossia da programmi che lavorano con la codifica utf8 per costruzione.

La data in piemontese è definita nelle righe da 33 a 38; come si vede la data piemontese non solo tratta i nomi dei mesi come nomi comuni, come in italiano, ma viene anche formata con l’ausilio di preposizioni semplici e articolate.

Le righe da 40 a 49 impostano alcuni parametri per la sillabazione; la sillaba iniziale e quella finale

non devono contenere meno di due caratteri; siccome moltissime parole piemontesi terminano con gruppi di consonanti, tocca ai pattern provvedere a che l'ultima sillaba contenga almeno una vocale.

Inoltre sono impostate le penalità per le righe vedove e le righe orfane; il valore 3000 è abbastanza alto ma non infinito, per cui le righe orfane e vedove potranno essere presenti ma *pdflatex* cercherà di evitarle con sufficiente determinazione. Il valore enorme assegnato a `\finalhyphendemerits` serve per evitare che la penultima riga di un capoverso subisca la cesura in fin di riga; in questo modo l'ultima riga di un capoverso non comincia (quasi) mai con una frazione di parola.

Le righe 50 e 53 impostano e disimpostano la condizione per la quale l'apostrofo possa essere considerato come parte di una parola in piemontese e non lo sia più quando si cambia lingua.

Le righe dalla 55 alla 84 sono la particolarità del piemontese; con l'occasione si sono risolti alcuni altri problemini di secondaria importanza, ma il cuore di questo codice serve per introdurre un carattere attivo (`"`)[3] (che è dichiarato attivo anche nella maggioranza dei file di descrizione delle lingue che si possono usare con LaTeX) al fine di poter gestire la doppia funzione del trattino, quella copulativa e quella diacritica.

Nelle sue funzioni di trattino copulativo LaTeX, per impostazione predefinita, gestisce la cosa in modo considerato tipograficamente corretto, cioè interdice la sillabazione delle due parole unite dal trattino, ma consente di andare a capo solo dopo il trattino; questa soluzione impedisce che entrino in gioco anche i trattini di cesura, cosicché non si stampano cose che potrebbero risultare ambigue.

La funzione diacritica è specifica del piemontese e serve per marcare il suono speciale (velare o faucale) da associare al "digrafo" 'n-'; serve anche per staccare la 's' da un nesso successivo del tipo 'ce' o 'ci', in mancanza del quale il lettore che conosce l'italiano penserebbe di dover pronunciare 'sce' e 'sci' all'italiana.

Pertanto nelle righe da 55 a 57 si dichiara che il segno `"` è attivo per la lingua piemontese e si predispone che questa caratteristica sia sempre attivata quando si compone in questa lingua; nelle righe da 85 a 86 queste proprietà sono disattivate nel momento in cui si cambia lingua.

Che cosa faccia il carattere attivo `"` è specificato nelle righe da 58 a 65: in modo matematico si deve comportare come ci si aspetta per la matematica (cioè serve per comporre un doppio apice) mentre in modo testo deve comportarsi come il comando interno `\pms@next`. A sua volta questo comando è definito come robusto nelle righe da 67 a 84; per prima cosa "azzera" il significato della macro di servizio `\pms@@next`; poi subordinatamente all'esito di certi test ridefinisce questa macro in modo acconcio, e infine questa macro viene eseguita.

I test servono per verificare se il token che segue il segno `"` è una lettera dell'alfabeto (escluse le lettere con diacritici), oppure se è una barra diritta, oppure se è un trattino, oppure se è un altro `"`. Nei quattro casi si hanno le seguenti funzionalità: (*a*) si inserisce un punto di possibile cesura in una parola, molto utile se la parola è composta e si desidera ottenere una divisione etimologica invece che fonetica; (*b*) inserisce una barra oppositiva che non impedisce la sillabazione delle due parole disgiunte dalla barra; (*c*) inserisce il trattino diacritico che non impedisce la sillabazione delle due parti di parola che il trattino unisce; (*d*) consente di inserire le doppie virgolette alte aperte che con la tastiera italiana sono generalmente difficili da introdurre. A noi evidentemente interessa la funzione del trattino diacritico.

Va notato che non è necessario usare sempre il segno `"` dopo la 'n' velare; si inserirà quel segno solo nel caso che, rivedendo le bozze di un documento, si noti che la sua mancanza produce una divisione in fin di riga errata.

Le ultime due righe completano le impostazioni per il piemontese e concludono il file; dopo `\endinput` potrebbe esserci scritta qualunque cosa, ma questa parte del file non verrebbe nemmeno letta da babel o polyglossia.

4.2 I file per la sillabazione

I file per generare le strutture interne per la sillabazione sono tre: `loadhyph-pms.tex`, `hyph-quote-pms.tex` e `hyph-pms.tex`; `pms` è la sigla del piemontese secondo le norme ISO.

Il primo file carica gli altri due quando si devono creare o ricreare i file di formato al momento di generarli, non durante la composizione di un particolare testo. Il secondo contiene le impostazioni per la codifica dell'apostrofo; infine il terzo file contiene i pattern veri e propri.

Il metodo che ho seguito per creare i pattern non si affida al programma *patgen* creato da Frank Liang, (Liang, 1983), e distribuito contestualmente con il sistema TeX, ma si affida alla traduzione nello speciale linguaggio (descritto da Knuth nel suo manuale *The TEXbook*, (Knuth, 1996)) delle regole grammaticali valide per il piemontese; queste regole sono certamente simili a quelle dell'italiano, ma non le ho trovate documentate da nessuna parte. Ho preso quindi delle decisioni in base alla mia esperienza, ma non è detto che queste decisioni siano corrette in ogni situazione; in particolare ho già accennato sopra ai problemi che nascono con il trattino copulativo o diacritico e con il digrafo 'ss'.

3. Usando XeLaTeX bisogna specificare la scelta della lingua piemontese o con `\setmainlanguage`, se è la lingua principale del documento, oppure `\setlanguage`, se si tratta di una lingua secondaria, ma in entrambi i casi bisogna specificare l'opzione babelshorthands per attivare questo segno.

Per fortuna la sillabazione tipografica ha certe sue regole più restrittive delle regole grammaticali, per cui ho semplicemente evitato le divisioni di vocali che *potrebbero formare* dittonghi o trittonghi. È noto, infatti, che il lettore si trova a disagio se, trovando una cesura in fin di riga, trova nella riga successiva un moncone di parola che comincia con una vocale[4]; perciò non si sono mai divisi gli iati; l'approccio che ho usato è: "meglio una divisione mancata che una divisione sbagliata". Inoltre il piemontese usa molto gli accenti tonici (anche nella funzione di accenti fonici); ho evitato di definire pattern contenenti vocali accentate, cosa che ho fatto in quasi tutti i file di pattern che ho creato per diverse lingue. Uno dei vantaggi di questo approccio è che il file dei pattern contiene solo caratteri a 7 bit, cioè ASCII puri.

Per il resto le regole che ho seguito sono le seguenti:

- Ogni sillaba contiene una vocale o un dittongo o un trittongo; in piemontese le vocali possono essere accentate; i dittonghi e i trittonghi sono di più di quelli presenti in italiano, grazie al ruolo di vocale chiusa che ha 'o' in piemontese e alla semivocale 'j'. Con lo stesso concetto dei pattern per l'italiano ho scritto i pattern per il piemontese evitando di esplicitare le vocali, cosicché, eventualmente, non vengono divisi alcuni iati, ma non vengono commessi errori dividendo gruppi vocalici dove non è consentito.

- Ogni consonante semplice e ogni digrafo seguito da una vocale o da un dittongo fa sillaba con la vocale che segue. I digrafi che corrispondono a "suoni semplici" sono: *cc, ch, gg, gh, gn, qu, ss*, ma ho preferito trattare il digrafo 'ss' come una consonante doppia; i digrafi 'cc' e 'gg' compaiono generalmente in fine di parola, e quando sono interni solitamente sono seguiti dal trattino; in questi rari casi, se e soltanto si presentano inconvenienti di sillabazione, il compositore si preoccuperà di usare il carattere attivo " per 'correggere' la sillabazione in fin di riga.

- Ho deciso che due consonanti diverse *che non formino un digrafo*, si dividono fra le sillabe adiacenti se la coppia di consonanti non può trovarsi all'inizio di una parola piemontese. Questo implica tra l'altro che le consonanti liquide 'l' ed 'r' e nasali 'm' ed 'n' vengono separate dalle consonanti che le seguono. La regola complessiva di questo punto è, *mutatis mutandis*, sostanzialmente uguale a quella per l'italiano.

4. Eccettuate le semivocali piemontesi 'i' e 'o', che marcano l'inizio di un dittongo o di un trittongo.

- È vietato andare a capo dopo l'apostrofo, esattamente come in italiano.

- Il trattino usato come "segno diacritico" non può venire gestito dai pattern. Mi è parso sufficiente definire il carattere attivo " per usalo in quei casi dove il trattino funziona da segno diacritico.

Con queste regole ho creato i pattern per il piemontese semplicemente prendendo i pattern per l'italiano, che già facevano abbastanza bene il loro lavoro, perché rispettano alcune regole abbastanza comuni alle lingue romanze, e ho poi aggiunto i pattern specifici per il piemontese, correggendo eventualmente quelli dell'italiano, quando erano in contrasto con le regole enunciate sopra. I pattern totali sono saliti a 374, 36 in più di quelli per l'italiano. Rigenerati i file di formato, sulla mia macchina ora posso scrivere in piemontese senza problemi. Spero che al momento di pubblicare questo articolo questa possibilità sia già stata integrata nelle distribuzioni ufficiali del sistema TEX.

Ecco ora un breve testo in piemontese tratto da Internet:

> I amëtto che a peul soné dròlo ocupesse 'd lenga piemontèisa ant un moment ëd globalisassion, andoa a smija che tuti a deubio mach studié le mideme còse an tut ël mond. Për mi contut (da bon bastian contrari...) a resta sempre valid ël prinsipi për ël qual na pianta ch'a l'ha nen ëd rèis a peul nen chërse e che donca a sia fondamental savèj da andoa as ven për savèj as va. E peui, second në studi che la Region Piemont a l'ha comissionà a l'Istitut d'arserche statistiche IRES ant ël 2006 a l'é vist-se che an Piemont su na popolassion d'apopré 4 500 000 përson-e, 2 000 000 a parlo ancora an lenga e n'àutr 1 300 000 a lo capiss bele s'a lo parla nen.[5]

5 Conclusione

Ho imparato molto da questo progetto dedicato alle lingue alpine; ho approfondito la mia conoscenza delle lingue romanze e ho scoperto realtà che prima mi erano assolutamente sconosciute. Anche in questo caso, come in quello degli articoli

5. Ammetto che può sembrare ridicolo occuparsi della lingua piemontese in un momento di globalizzazione, dove sembra che tutti debbano studiare solo le stesse cose in tutto il mondo. Nonostante ciò per me (da buon bastian contrario...) resta sempre valido il principio per il quale una pianta che non ha radici non può crescere e che dunque sia fondamentale sapere da dove si viene per sapere dove si va. E poi, secondo uno studio che la Regione Piemonte ha commissionato all'Istituto di ricerche statistiche IRES nel 2006, si è visto che in Piemonte su una popolazione di circa quattro milioni e mezzo di persone, due milioni parlano ancora in lingua e che un milione e trecentomila lo capiscono anche se non lo parlano.

precedenti, (Beccari, 2012) e (Beccari, 2013), al di là delle questioni specifiche di questa o quella lingua, ho rivisto bene quale sia il processo di gestione delle lingue gestito dai pacchetti babel e polyglossia. I file che ho predisposto sono già stati inviati ai gruppi di lavoro che amministrano questi pacchetti e spero che presto chiunque lo voglia, possa scrivere anche in piemontese.

6 Ringraziamenti

Di solito si ringrazia qualcuno; nel caso del piemontese non devo ringraziare una persona in particolare, ma vorrei ringraziare gli innumerevoli piemontesi che ho incontrato da quando sono immigrato da bambino in Piemonte e che mi hanno inculcato l'amore per la loro lingua; tra questi mi piace ringraziare coloro che mi hanno insegnato i primi rudimenti, gli amici Venanzio e Maresa, veri piemontesi delle terre del Barbaresco, che, quando costruivo casa qui dove ora abito, alla fine di una conversazione mi dissero: «E lei vuol venire ad abitare qui senza conoscere il piemontese?» Furono le ultime parole che mi dissero in italiano...

Riferimenti bibliografici

Alberti, L. B. (1973). *Grammatica della ligua toscana*, Laterza, volume III di *Collana "scrittori d'Italia"*. L'opera originale fu pubblicata attorno al 1442. Scaricabile in formato PDF dal sito `http://www.liberliber.it/libri/a/alberti/index.htm`.

— (1998). *De componendis cyfris*. Galimberti Tipografi Editori.

Alighieri, D. (1990). *De vulgari eloquentia*. Oscar classici. Mondadori, Milano.

Beccari, C. (2012). «LaTeX e le lingue romanze alpine». *ArsTEXnica*.

— (2013). «LaTeX e le lingue romanze alpine: il friulano». *ArsTEXnica*.

Braams, J. (2011). *Babel, a multilingual package for use with LaTeX's standard document classes*. TUG. Leggibile con `texdoc babel` nella distribuzione TeX Live.

Brero, C. (2008a). *Gramatica e sintassi dla lenga piemontèisa*. Editrice Il Punto – Piemonte in Bancarella, Torino. La prima versione di questa grammatica è stata pubblicate nel 1967.

— (2008b). *Nuovo Vocabolario italiano-piemontese e piemontese-italiano*. Editrice Il Punto – Piemonte in Bancarella, Torino. Ristampa della prima edizione del 1967.

Cavallotto, Allasia, Buonanno, Fogliatto, Pastore, Simonetti e Togni (2011). «Modifica all'articolo 2 della legge 15 dicembre 1999, n. 482, in materia di riconoscimento, tutela, e valorizzazione del patrimonio linguistico, risorgimentale, letterario e filologico della lingua regionale piemontese». `http://http://parlamento.openpolis.it/atto/documento/id/61377`.

Charette, F. (2011). *Polyglossia: a Babel replacement for XeLaTeX*. TUG. Leggibile con `texdoc polyglossia` nella distribuzione TeX Live. L'attuale versione è affidata alla manutenzione di Arthur Reutenauer.

Chiesa Valdese (1834). *'L Testamente Neuv dë Nossëgnour Gesu-Crist*. Moyes, Londra.

Delfuoco, S. e Bernardi, P. (a cura di) (2004). *Sermoni Subalpini del XII secolo*. Centro di Studi Piemontesi, Torino. Trascrizione a cura di Giuliano Guasca Queirazza. Riproduzione integrale del manoscritto a colori con introduzione di Letizia Sebastiani.

Fukui, R. (2004). *TIPA manual*. Graduate School of Humanities and Sociology, Tokyo University, Tokyo. Leggibile con `texdoc tipaman` nella distribuzione TeX Live.

ISO (2007). *ISO 639-3:2007 – Codes for the representation of names of languages*. International Standards Organisation, Ginevra.

Knuth, D. E. (1996). *The TEXbook*. Addison Wesley, Reading, Mass., 16ª edizione.

Liang, F. M. (1983). *Word Hy-phen-a-tion by Com-puter*. Tesi di Dottorato, Department of Computer Science, Stanford, CA. URL `www.tug.org/docs/liang/liang-thesis.pdf`.

Pipino, M. (1783). *Gramatica Piemontese*. Reale Stamperia. Copia anastatica pubblicata dalla Ca dë Studi Piemontèis di Torino, con il patrocinio della Regione Piemonte, nel 2006.

▷ Claudio Beccari
Villarbasse
`claudio dot beccari at gmail dot com`

LaTeX e la documentazione di progetto

Roberto Giacomelli

Sommario

In questo articolo verranno esaminati vantaggi e svantaggi dell'uso del sistema TeX per la composizione della documentazione di progetto nel settore professionale degli ingegneri, per esempio nei campi civile, industriale ed elettronico.

Numerose caratteristiche di questo tipo di documentazione producono requisiti tipografici e di coerenza stilistica, nonché un problema generale di affidabilità. Saranno quindi presentate alcune soluzioni derivate dall'esperienza quotidiana di confronto con gli altri programmi di impaginazione.

Abstract

In this paper I will examine advantages and disadvantages of the TeX system used in professional engineering fields, such as the civil, the industrial or the electronic ones, to typeset a variety of project documents.

Plenty of specific features of this kind of documentation produce typographic and style coherence requirements, along with a general problem of reliability. Some solutions suggested by the everyday personal experience will be presented and compared to those allowed by other software programs.

1 Obiettivi dell'articolo

I punti chiave che si vogliono definire e sviluppare in questo articolo sono i seguenti:

- segnalare ai tecnici progettisti l'esistenza del sistema TeX come strumento utile per la composizione della documentazione dei progetti d'ingegneria;
- illustrare brevemente le peculiarità ed i requisiti documentali che si affrontano durante il lavoro, suddividendoli in tre categorie: *tipografici*, *di affidabilità* e *gestionali*;
- presentare le potenzialità del sistema TeX nei confronti dei requisiti citati;
- presentare esempi di documenti tratti da progetti reali redatti nel formato LaTeX.

La documentazione del progetto d'ingegneria costituisce un argomento importante e assai vasto; implica aspetti *giuridici* per la responsabilità dei tecnici, *normativi* nei confronti di precise prescrizioni sui contenuti e coinvolge le procedure di qualità come la conservazione e l'identificazione dei documenti digitali.

Gli obiettivi saranno perseguiti non nell'intento di esaudire tutte le questioni citate ma dando risposte tratte dall'esperienza del lavoro svolto nell'attività di progettazione.

2 Progetto ed informatica

Con l'estendersi degli strumenti informatici e delle comunicazioni consentite dalla rete internet, i documenti di progetto sono oggi prodotti esclusivamente nel formato digitale. Si è passati in una quarantina d'anni dal regolo calcolatore ai modelli agli elementi finiti, dalla scrittura manuale o con la macchina da scrivere all'invio di file verso perfezionati sistemi di stampa a colori.

Questa tendenza è destinata ad evolversi ancor di più nel futuro, modificando la progettazione in un'attività sempre più vicina alla simulazione, tramite il supercalcolo dei fenomeni reali nei settori industriale, civile ed elettronico (Myers, 2013).

Nello scenario di estesi gruppi di ingegneri, che utilizzano potenti calcolatori multiprocessore per sofisticate modellazioni numeriche, riuscirà il sistema TeX, ideato da Donald Knuth nel 1978 e con oltre tre decenni di onorato servizio, ad essere completamente a proprio agio nell'esaudire le richieste di composizione di documenti complessi e soggetti a particolari cicli di sviluppo?

Questo articolo vuole richiamare l'attenzione degli utenti professionali del settore dell'ingegneria sulla necessità di una corretta e consapevole scelta del sistema adottato in questo tipo di lavori di documentazione.

3 Documentazione di progetto

Un progetto d'ingegneria consiste in un insieme di documenti che contengono le analisi, la descrizione, le modalità di costruzione ed il modo d'uso di un'opera o di un manufatto.

Molti dei documenti che compongono un progetto sono relazioni tecniche o rapporti numerici dei modelli matematici, che assumono la forma di fascicoli stampati e rilegati in un ridotto numero di copie. Spesso possono assumere una valenza giuridica anche di rilevanza penale, come accade per la documentazione depositata presso gli uffici pubblici per il rilascio delle autorizzazioni previste per legge nei vari ambiti dell'ingegneria.

In questa sezione descriveremo le caratteristiche di questi documenti nonché delle necessità prati-

che degli autori; nella prossima sezione descriveremo come gli strumenti del sistema TEX possono facilitarne la stesura, assicurando al processo di redazione affidabilità e sicurezza.

3.1 Caratteristiche tipografiche

La documentazione tecnica di un progetto è caratterizzata dai seguenti aspetti tipografici e compositivi:

- necessità della perfetta uniformità grafica e coerenza tipografica tra *tutti* i documenti del progetto;
- necessità di un frontespizio conforme ad uno standard, di indici generali e di indici per tabelle, figure e listati di codice, nonché della completa numerazione ed identificazione delle pagine;
- necessità di produrre documenti anche molto complessi per struttura, qualità e molteplicità dei contenuti; per esempio quando contengono schemi al tratto, grafici di funzione o sperimentali, tabelle anche molto estese, testo ricco di riferimenti normativi e bibliografici, matematica, eccetera.

3.2 Affidabilità dei contenuti

Per affidabilità dei contenuti s'intende la corretta corrispondenza del testo inserito dall'autore con ciò che egli intendeva effettivamente esprimere.

Per esempio, molto spesso gli indici dei documenti generati dai programmi di *word processing* contengono riferimenti errati od assenti, oppure le formule matematiche lasciano il rischio di essere lette in modo errato.

Nell'ambiente di lavoro al computer, ciò che favorisce l'automazione dei riferimenti e degli indici e la corretta composizione del testo o delle formule favorisce l'affidabilità dei contenuti, oltre che il lavoro di stesura. Di seguito è riportato un elenco di punti chiave per i documenti dei progetti:

- necessità di una sicura e completa identificazione dei documenti e della versione (per esempio *valida*, *superata* o *controllata*);
- necessità di indici e riferimenti accurati ed affidabili;
- necessità di chiarezza nella composizione della matematica, per evitare fraintendimenti nelle formule;
- conformità a norme nazionali od internazionali dei contenuti ed in particolare a quelle relative alle unità di misura;
- necessità di facilitare il controllo e la revisione per minimizzare il numero degli errori;
- massima chiarezza e concisione del testo.

3.3 Gestione informatica

Se, come accennato in apertura nella sezione 2, la documentazione di progetto è oggi prodotta con l'elaborazione digitale, nascono alcune importanti questioni come la sicurezza dei dati, la stabilità del sistema e la facilità di memorizzazione.

Ecco un elenco di alcuni punti importanti relativi agli aspetti di una buona gestione digitale:

- vengono richiesti e utilizzati dati comuni a più documenti, come per esempio il titolo del progetto ed i nominativi dei tecnici responsabili;
- nei diversi progetti, interi brani di testo si ripetono uguali o subiscono limitate modifiche;
- i file digitali devono essere archiviati in copie di backup e conservati digitalmente. Occorre poi che siano gestiti con criteri di sicurezza e controllo delle revisioni;
- necessità di lavoro di gruppo e di scambio della documentazione.

4 TEX nel progetto

Le capacità di composizione del sistema TEX consentono di ottenere un'elevata qualità tipografica. Questi programmi di composizione operano infatti con una modalità detta *asincrona*, elaborando il materiale da comporre senza i vincoli tipici dei word processor, dovuti alla necessità di presentare il documento finale a schermo in tempo reale.

Tuttavia, la riconosciuta qualità tipografica di TEX — offerta per di più senza costi di licenza — non è l'unica caratteristica essenziale per gli scopi della documentazione di progetto: occorre la capacità di rispondere in modo adeguato all'esigenza primaria di minimizzare gli errori.

La buona tipografia favorisce la lettura, la coerenza di stile, la chiarezza della struttura logica del documento e per questo tende a ridurre gli errori, favorendo una maggiore attenzione durante la stesura del testo ed una maggiore precisione di controllo delle bozze.

Nei prossimi paragrafi, per ciascuna delle necessità documentali espresse nella sezione precedente, si elencheranno i metodi operativi implementabili con il sistema TEX ed in particolare con il formato LATEX.

4.1 Coerenza stilistica

La coerenza tipografica può essere raggiunta, in modo pressoché perfetto ed automatico, scrivendo una o più classi di documento per il formato LATEX, al cui interno inserire le impostazioni di pagina, i comandi di frontespizio e quelli per la composizione delle firme dei responsabili.

Utilizzando i file di classe, diviene possibile apportare modifiche e miglioramenti mano a mano

che cambiano le esigenze e di perfezionare il codice stesso di pari passo con la crescita delle proprie conoscenze TEX, fino ad arrivare alla programmazione più sofisticata, per esempio con il linguaggio LATEX3 o con LuaTEX.

L'uso di classi documento personalizzate non è solo una soluzione *naturale* ma permette anche di creare una *gerarchia* di classi sempre più specializzate ed adatte per il singolo tipo di documento: è possibile infatti derivare una classe a partire da un'altra inserendo il codice comune a tutti i documenti nella classe base. Una modifica nella classe base si rifletterà automaticamente in tutte le altre, senza che siano necessarie ripetizioni per il singolo tipo di documento (lettera, relazione o fascicolo di calcolo).

Il consiglio che mi preme dare sull'argomento è il seguente: scrivere e tenere aggiornati i manuali delle proprie classi di documento, in modo tale da ritrovare la sintassi delle macro senza perdite di tempo. Questi riferimenti diventano essenziali se si lavora in gruppo allo stesso progetto.

Per comprende come scrivere una classe di documento, è possibile consultare la guida tematica del GUIT disponibile nella sezione Documentazione del sito dell'associazione (Beccari, 2013).

4.2 Complessità del documento

I processi di compilazione del sistema TEX, essendo eseguiti su file sorgenti, risultano essere in grado di elaborare correttamente documenti anche molto complessi di centinaia di pagine; per esempio, lunghi elaborati numerici tabellari composti su un numero variabile di colonne non sono un problema, cosa invece molto impegnativa per un word processor perché il programma deve rigenerare ad ogni piccola modifica tutto il documento per presentarlo composto a schermo.

Tuttavia, è necessario che l'utente rispetti le necessarie sequenze di compilazione previste per il completamento delle bibliografie, degli indici e per la risoluzione dei riferimenti. I documenti di progetto riportano normalmente la dicitura "pagina $\langle n \rangle$ di $\langle tot \rangle$", perciò come minimo devono essere previsti due passaggi di compilazione per ottenere il documento finale completo. Le informazioni trascritte nei file di *log* e nei messaggi di compilazione in console aiutano gli utenti a stabilire nella sequenza delle compilazioni quando sono necessarie ulteriori elaborazioni.

Se il documento contiene molti oggetti mobili — le tabelle e le figure che vengono posizionate automaticamente nel documento senza che l'utente debba preoccuparsi di sistemare manualmente il testo — la compilazione richiede un po' di tempo in più. Come riferimento, un documento di un centinaio di pagine con circa 150 oggetti mobili richiede una trentina di secondi su un sistema hardware che oggi viene classificato come mediamente veloce.

È anche vero che durante la stesura del documento è facile limitare la composizione ad una parte molto più piccola del documento, per esempio ad un solo capitolo — e magari utilizzare un file di formato personalizzato per diminuire ulteriormente i tempi di caricamento dei vari pacchetti speciali — così che l'elaborazione sia sufficientemente veloce da non disturbare la fase di stesura[1].

4.3 Grafici vettoriali

Poiché il formato PDF è direttamente supportato dal più utilizzato programma di composizione `pdftex`, è possibile inserire nei documenti file in quel formato realizzati con i programmi esterni CAD.

I recenti programmi CAD supportano infatti il plottaggio diretto verso file PDF e spesso sono rilasciati con i driver per l'uscita nel formato PostScript, facilmente convertibile in PDF. In ogni caso, è possibile ricavare qualsiasi disegno occorra ed inserirlo nella relazione come normale oggetto grafico, variandone eventualmente scala ed angolo di rotazione.

Non è più un problema inserire nella relazione disegni perfettamente in scala con la qualità vettoriale di un particolare costruttivo o di una planimetria. La grafica vettoriale può essere prodotta anche direttamente all'interno del sorgente, utilizzando potenti linguaggi messi a disposizione da pacchetti per LATEX o per altri formati come ConTEXt. Una galleria di esempi di grafica programmata completa di codice sorgente può essere consultata nel sito dedicato al pacchetto TikZ, alla pagina `http://www.texample.net/tikz/examples/`.

Grazie al pacchetto pdfpages è poi molto semplice inserire allegati in formato PDF nel documento principale, per esempio per aggiungere le specifiche e le validazioni di un software di calcolo direttamente dalla documentazione ufficiale del programma o certificazioni di prodotto.

Per le immagini *raster* si consiglia di utilizzare il formato PNG (estensione `.png`) se l'immagine contiene un ridotto numero di colori, come per esempio per quelle delle schermate grafiche come finestre di *screenshot*, ed il formato JPEG (estensione `.jpg`) per il materiale fotografico, regolandone opportunamente la risoluzione in funzione di quella del dispositivo di stampa o delle modalità d'uso a video del documento.

4.4 Correttezza dei riferimenti

Con le classi standard di LATEX l'indice generale è composto in modo automatico dal comando

```
\tableofcontents
```

1. Nel post `http://www.guitex.org/home/it/forum/5-tex-e-latex/68351-creare-un-file-di-formato-personalizzato` sul forum del GUIT si può leggere una discussione nel merito all'uso dei formati precompilati per abbreviare i tempi di compilazione.

che elaborerà le informazioni memorizzate in un file ausiliario dai comandi di sezionamento come `\section{}` e `\chapter{}`.

Come già evidenziato, è indispensabile procedere con il giusto numero di passaggi di compilazione per ottenere l'indice completo con gli esatti riferimenti[2].

Nel testo possono essere inseriti i riferimenti — anche ipertestuali[3] — di capitoli e sezioni, formule matematiche, tabelle e figure, siti web e rimandi bibliografici.

4.5 Correttezza della matematica

Tra i punti di forza di TEX c'è la sua formidabile capacità di comporre la matematica. Il risultato consente al lettore di evitare errori di interpretazione dovuti all'errato posizionamento di linee di frazione o di esponenti.

Per esempio la relazione seguente è così riportata in un testo normativo:

$$S_\mathrm{e}(T) = a_\mathrm{g} S \eta F_\mathrm{o} \{[(T/T_\mathrm{B}) + [1/(\eta F_\mathrm{o})] \cdot [1 - (T/T_\mathrm{B})]\}$$

ma è certamente più chiara e quindi meno soggetta ad errori d'interpretazione se viene scritta nella forma

$$S_\mathrm{e}(T) = a_\mathrm{g} S \eta F_\mathrm{o} \left[\frac{T}{T_\mathrm{B}} + \frac{1}{\eta F_\mathrm{o}} \left(1 - \frac{T}{T_\mathrm{B}} \right) \right]$$

4.6 Unità di misura

Spesso nelle formule e nel testo occorre inserire grandezze fisiche dotate di unità di misura. Il pacchetto siunitx per LATEX è molto utile per comporre le unità di misura perché specifico del sistema internazionale SI.

Ecco un brano di esempio tratto da una relazione tecnica:

> Per il sito di costruzione in zona II con altitudine $a_\mathrm{s} = 217\,\mathrm{m}$ e valore caratteristico dell'azione della neve:
>
> $$q_\mathrm{sk} = 0{,}85 \left[1 + \left(\frac{a_\mathrm{s}}{481} \right)^2 \right] = 1{,}02\,\mathrm{kN/m^2}$$
>
> l'azione di progetto della neve vale $0{,}82\,\mathrm{kN/m^2}$. Infatti, con C_t e C_E di valore unitario e μ_i pari a 0,80 per falde di copertura a due falde con inclinazione $\alpha < 30°$ risulta:
>
> $$q_\mathrm{s} = \mu_\mathrm{i}\, q_\mathrm{sk}\, C_\mathrm{E}\, C_\mathrm{t} = 0{,}82\,\mathrm{kN/m^2}$$

Il codice LATEX corrispondente è il seguente (l'indentazione in ambiente matematico è utilizzata per facilitare il controllo della struttura algebrica delle espressioni; le unità di misura sono composte con le stesse macro di siunitx sia in ambiente matematico sia in modo testo):

```
Per il sito di costruzione in zona
\textsc{ii} con altitudine
\(a_\mathrm{s} = \SI{217}{m}\) e valore
caratteristico dell'azione della neve:
\[
q_\mathrm{sk} =
   \num{0.85}
   \left[
      1+\left(
           \frac{a_\mathrm{s}}{481}
        \right)^2
   \right]
   =
   \SI{1.023}{kN/m^2}
\]
l'azione di progetto della neve vale
\SI{0.82}{kN/m^2}.

Infatti, con
$C_\mathrm{t}$ e $C_\mathrm{E}$
di valore unitario e $\mu_\mathrm{i}$
pari a \num{0.80} per falde di copertura
a due falde con inclinazione
$\alpha<\ang{30}$ risulta:
\[
q_\mathrm{s} =
   \mu_\mathrm{i}\, q_\mathrm{sk}\,
   C_\mathrm{E}\, C_\mathrm{t}
   =
   \SI{0.82}{kN/m^2}
\]
```

4.7 Chiarezza del testo

LATEX favorisce la chiarezza espositiva per un paio di fattori: si lavora all'interno di un editor di testo, senza distrazioni come quella di controllare tipo e dimensione del font in uso; inoltre la qualità tipografica del documento può invogliare gli autori ad assumere un metodo di miglioramento continuo dei propri mezzi espressivi.

Ci sono molti testi sull'argomento piuttosto vasto della scrittura tecnica e scientifica, come per esempio quello liberamente scaricabile del Politecnico di Torino (Beccari *et al.*, 2013), ed anche direttive di carattere normativo che individuano le caratteristiche principali da tener in considerazione per produrre un testo chiaro, leggibile e senza fronzoli.

Non è poi così facile ottenere un testo pulito e ben strutturato. Occorre allenamento ed un buon lavoro di revisione. Nella rivista dell'UNI (Erika Leonardi, 2013), si legge che "*una frase ben leggibile in tutti i contesti è formata mediamente da 25 parole: è il suggerimento applicato come esempio reale e concreto per convincervi!*"

Di sicuro scrivere articoli per la rivista del gruppo (ArsTEXnica) è un ottima occasione di miglioramento — e non occorre essere degli esperti ma anche solo raccontare le proprie esperienze con TEX.

2. Per la personalizzazione dello stile dell'indice possiamo affidarci al pacchetto titletoc
3. Il pacchetto che inserisce questi riferimenti è hyperref.

4.8 Sicurezza dei file

Le informazioni che servono ai motori di composizione del sistema TEX per produrre i documenti finali si trovano all'interno di file di testo chiamati *sorgenti*.

La prima conseguenza è che i sorgenti non sono modificati dai programmi di composizione perché utilizzati solo in lettura. Questo comporta che il software tipografico non può corrompere o danneggiare le informazioni anche se contenesse errori di programmazione. Tutt'al più non funzionerebbe.

La seconda osservazione sulla sicurezza dei file riguarda la conservazione dei documenti digitali. Sappiamo infatti che all'interno dei file i nostri dati sono codificati in uno specifico formato di cui tuttavia non è possibile conoscere niente. Il problema è trattato per esempio nel brillante articolo Rothenberg (1995) e consiste nel fatto che descrivere digitalmente il formato di registrazione comporta ricorsivamente la necessità di un formato in cui codificare le specifiche del formato iniziale. In altre parole, l'informazione binaria di per sé non può mai essere completa e la perdita del programma originario, in grado di riprodurre il file, comporta l'impossibilità di risalire ai dati dalla loro rappresentazione digitale.

Tuttavia, proprio come fareste voi con un file di estensione sconosciuta giunto nella vostra casella di posta elettronica, il primo tentativo per scorgere il contenuto digitale è quello di aprirlo con un editor che utilizza la codifica ascii.

Per i sorgenti la codifica può essere scelta dall'utente ma, poiché il formato ascii non comprende i caratteri accentati, la cosa migliore da fare è utilizzare invece quella che di fatto ne è un'estensione: *Unicode*, in particolare nella variante chiamata `utf-8`. Molti sistemi operativi ed editor di testo riconoscono questo standard internazionale permettendo di lavorare sugli stessi file praticamente ovunque.

Oggi, anche i word processor salvano i file come dati testuali in un formato compresso — xml con la compressione `zip` nel caso del formato *Open Document Format* delle suite Apache Open Office e Libre Office. La compressione diminuisce le probabilità che nel futuro i file possano ancora rivelare l'informazione contenuta.

Riassumendo:

- i sorgenti vengono elaborati in sola lettura dai motori di composizione ed in lettura/-scrittura dagli editor, minimizzando i rischi di alterazione accidentale;
- il formato dei sorgenti è testo puro preferibilmente in codifica Unicode utf8, che ha ottime probabilità di rimanere leggibile per molti anni;
- i file di testo dei sorgenti sono elaborati in lettura/scrittura da molti diversi editor praticamente su tutte le piattaforme hardware e software per la massima portabilità e condivisione;
- i file di testo sono facilmente elaborati dai programmi per il *controllo delle revisioni* che possono funzionare senza problemi anche in archivi di rete.

4.9 Documenti archiviabili

Dalla sezione precedente sappiamo che il modo forse più sicuro per mantenere la sicurezza dell'informazione a lungo termine è quello di utilizzare file in formato testo. I sorgenti soddisfano pienamente questo formato ma non riproducono in modo completo l'informazione contenuta nei documenti. Basti pensare alle immagini ed alla fedele riproduzione delle pagine.

Per questo scopo è stata emanata un'apposita norma, la iso 19005-1, che stabilisce le specifiche di un formato digitale — il formato pdf/a — nel tentativo di consentire ai file digitali conformi alle regole di riprodurre nel tempo il documento, che perciò diventa *archiviabile*.

Più nel dettaglio, il formato pdf/a della norma corrisponde al formato pdf 1.4 e, soprattutto, ad una serie di regole supplementari che riguardano per esempio i font ed i colori. Con il sistema TEX è possibile produrre documenti nel formato pdf archiviabile seguendo le procedure descritte, tra gli altri, negli articoli Beccari (2009), Scarso (2010) e Beccari (2011). Gli articoli citati dimostrano che l'argomento della conservazione dei documenti digitali è complesso. Spesso i programmi non riescono a soddisfare pienamente le specifiche del formato pdf/a — cosa che invalida l'intero documento — e per scoprirlo occorre usare del software specifico.

Il sistema TEX è oggi in grado di produrre documenti archiviabili sempre con maggiore facilità grazie al procedere della ricerca e dello sviluppo di soluzioni. Questa caratteristica diviene sempre più importante dal momento che la pubblica amministrazione tende a convertire al digitale la gestione documentale.

4.10 Backup e controllo di revisione

L'operazione di *backup* consiste essenzialmente nel copiare l'insieme di file di un progetto su un altro dispositivo di memorizzazione rispetto a quello in cui si trova la relativa cartella di lavoro. Il backup ha due obiettivi: salvaguardare i dati da problemi del dispositivo fisico di memoria e realizzare a lungo termine la conservazione dei documenti.

Invece di illustrare il funzionamento di software specifici per il backup, ci soffermeremo sui programmi per il *controllo di revisione*, il cui acronimo è vcs (*Version Control Systems*), dedicati in particolare alla gestione dei file di testo poiché progettati per lo sviluppo del codice sorgente.

Utilizzando un VCS con i sorgenti della documentazione di progetto, realizzeremo una sorta di sofisticato backup incrementale dei dati. Durante la progettazione è frequente l'abbandono di una soluzione o di un modello per un'alternativa — per esempio perché sono nel frattempo cambiate le specifiche da soddisfare. "Fotografare" tutte le versioni dei file di un progetto equivale dunque a costruire una sorta di albero, con rami abbandonati o ricchi di versioni, che ricalca esattamente il modo con il quale i progettisti lavorano:

- aggiungendo contenuto, cioè creando una nuova versione del progetto;
- apportando una modifica importante o ricominciando da capo, dunque creando un nuovo ramo;
- recuperando interamente o parzialmente una vecchia versione.

Il controllo di revisione permette di mantenere tutte le versioni ed i percorsi di sviluppo del lavoro digitale. La gestione di questo albero è manuale e demandata agli utenti. Essi, poiché i file sono in formato testo, possono in ogni momento confrontare due versioni con funzioni automatiche che riconoscono le modifiche apportate.

Ho gestito con successo tutta la documentazione di progetti — specie per quelli complessi — con il programma `git` creato da Linus Torvalds appositamente per lo sviluppo del kernel di Linux[4], mantenendo un repository locale sincronizzato con una copia su server di rete. Per saperne di più, esiste la *Guida all'uso di Git per gli utenti di LaTeX*, che non ha ancora raggiunto le altre guide tematiche, sviluppata a sua volta con lo stesso sistema da Mosè Giordano da un'idea iniziale di Pietro Giuffrida (indirizzo web `https://github.com/giordano/guidagit`).

Un altro esempio pratico di questo tipo di gestione è il *repository* del GuIT consultabile all'indirizzo web `https://github.com/GuITeX` dove vengono mantenuti i sorgenti delle guide tematiche e gli altri progetti a sviluppo aperto del gruppo.

4.11 Lavoro di gruppo

Con l'adozione di un sistema VCS descritto alla sezione precedente, si è in grado di operare sui documenti del progetto non solo in più persone ma anche da diversi luoghi.

Da solo il sistema TEX rende già piuttosto semplice il compito di organizzare il lavoro di gruppo: è possibile infatti suddividere il sorgente in più file, creando un sorgente *master* che includa gli altri file con appositi comandi. In questo modo, si può lavorare contemporaneamente allo stesso documento, ognuno con il proprio argomento e gestendo opportunamente i backup.

4. Si consulti il ricchissimo sito ufficiale di `git` all'indirizzo `http://git-scm.com/`.

Se si utilizzano cartelle di rete in cui condividere i sorgenti, non è possibile che più persone lavorino contemporaneamente e in sicurezza allo stesso file, perché uno non vedrebbe le modifiche dell'altro annullandole al momento del salvataggio del file dalla sua postazione: si dovrebbe ricorrere a preavvisi o ancora ad un sistema VCS professionale.

Il lavoro di gruppo può quindi procedere in modo efficiente e senza errori di gestione. Occorre tuttavia tener presente il problema dell'interoperabilità dei dati da e verso i word processor che si presenta solo se i sorgenti devono essere distribuiti o, viceversa, se i documenti devono essere importati. I programmi convertitori oggi disponibili potrebbero essere sufficienti, ma in generale la procedura non è ancora automatica. Ciò rappresenta un serio problema, specie se i due formati sono utilizzati all'interno della stessa organizzazione al lavoro su progetti comuni. Non esiste una soluzione automatica se non quella di poter disporre di convertitori efficaci.

5 Un confronto pagina per pagina

Recentemente ho svolto l'attività di ispettore per la validazione di un progetto pubblico per la costruzione di un edificio civile composto dal progetto delle strutture, dalle verifiche termiche per il risparmio energetico e dal progetto degli impianti.

I numerosi documenti del progetto sono stati redatti con programmi di word processing da tecnici diversi nell'ambito dello stesso studio professionale. Una situazione questa ideale per l'utilizzo di un sistema di composizione in grado di soddisfare le specifiche esigenze documentali e lo è anche per confrontare i risultati tipografici ottenuti.

Il confronto deve evidenziare non tanto le differenze tecniche, perché certamente anche con i word processor si possono inserire formule e testatine coerenti, ma valutare gli effetti delle modalità operative che favoriscono o meno la qualità dei contenuti.

Da un altro punto di vista, per realizzare lo stesso "oggetto tipografico" — per esempio un'espressione matematica numerata con applicazione numerica — occorre impegno con entrambi i sistemi. A parità di risultato campione, il confronto dovrà valutare quanto lavoro è necessario svolgere da utenti esperti in ciascuno di essi.

Ecco tuttavia cosa ho tipograficamente rilevato nelle relazioni di quel progetto da validare:

- problemi nella correttezza degli indici: a volte non solo il numero di pagina non corrisponde ma nemmeno il titolo della sezione. Ciò dà credito all'ipotesi che gli indici siano stati costruiti manualmente;
- il testo cambia di dimensioni ed interlinea nello stesso documento;

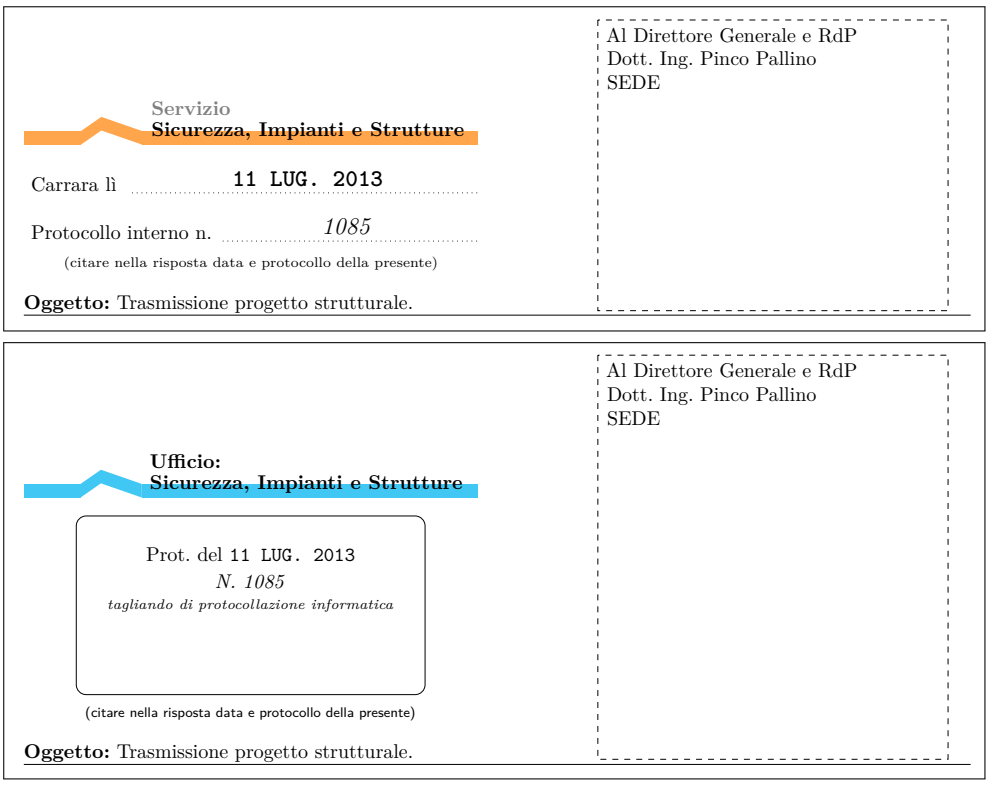


FIGURA 1: Porzioni centrali della prima pagina della lettera aziendale standard con il tag di protocollo ed il box degli indirizzi dei destinatari, qui evidenziato da un rettangolo tratteggiato. La versione superiore è quella interna mentre quella inferiore si riferisce allo stile delle lettere indirizzate all'esterno. Per impostare uno dei due stili è prevista un opzione di classe documento che consiste nella chiave `internal`.

- a volte le formule sono composte come testo normale, per esempio "Vr=-Tr x ln(1-Pv)" invece di $V_\mathrm{r} = -T_\mathrm{r} \log(1 - P_\mathrm{v})$ [5]. Altre volte sono composte correttamente ma sono prive di numerazione, altre volte sono immagini di tipo raster;
- le figure non hanno né indice né didascalia. La descrizione si trova invece nel testo;
- le unità di misura sono spesso non conformi al sistema internazionale SI, per esempio si legge 2,53 kg/cmq;
- le tabelle hanno stili diversi ed a volte sbordano clamorosamente dalla pagina, avendo colonne smisuratamente larghe rispetto al contenuto;
- i documenti non sono tipograficamente coerenti tra loro, probabilmente perché redatti singolarmente da tecnici.

6 Esempi

In questa sezione verranno illustrati esempi reali di documentazione prodotta con il sistema TEX riguardante progetti di ingegneria civile.

5. A ben vedere, la formula corretta riportata in normativa nel D.M. 14/01/2008 «Norme tecniche per le costruzioni», è invece la seguente: $V_\mathrm{r} = -T_\mathrm{r} / \log(1 - P_\mathrm{v})$.

Per chiarire, fino ad ora si è parlato di sistema TEX accennando al formato LATEX. Essenzialmente, i programmi di composizione della famiglia — detti anche *motori* — elaborano i file sorgenti per produrre i documenti finali, per esempio nel formato PDF. I motori di composizione elaborano un linguaggio di base con cui è possibile e conveniente definire nuovi comandi — che nell'insieme costituiranno un linguaggio solitamente chiamato *formato* — che costituiscono, per così dire, l'interfaccia utente del sistema. LATEX è quindi un linguaggio ad alto livello basato su TEX.

Alle macro del formato in uso possono aggiungersi funzionalità specifiche. Queste ultime possono essere contenute nei *pacchetti d'estensione* caricati esplicitamente nel sorgente con l'apposita istruzione oppure possono essere definite all'interno del sorgente stesso.

I successivi esempi sono stati tutti ottenuti da sorgenti scritti nel classico formato LATEX 2_ε, utilizzando vari pacchetti d'estensione, tra cui amsmath per la matematica, siunitx per la composizione di numeri ed unità di misura e graphicx per l'inclusione di immagini.

6.1 Una lettera aziendale

I suggerimenti dati alla sezione 4.1 sono stati messi in pratica per produrre lettere aziendali attraverso una classe di documento LATEX derivata dalla classe base article. Per motivi di riservatezza non è

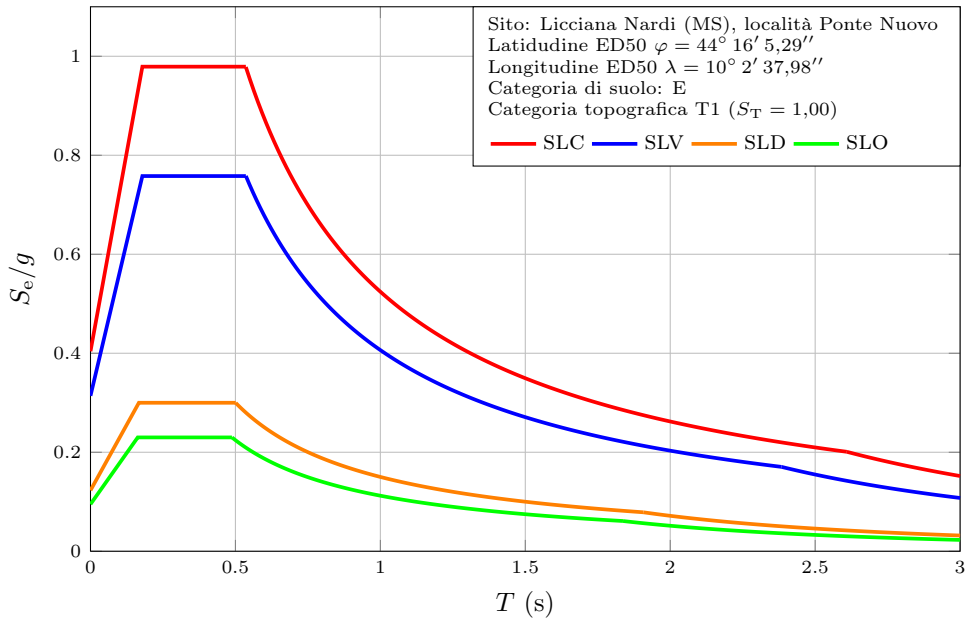


Figura 2: Esempio reale di grafico inserito in una relazione tecnica che riporta gli spettri elastici di progetto relativi ad un sito di costruzione. Queste curve sono molto importanti per la definizione del rischio sismico e per determinare la conseguente risposta strutturale.

possibile qui mostrare un'immagine intera di una lettera reale, ma solo spiegare come sono stati risolti i dettagli tipografici principali ed i possibili sviluppi in direzione dell'utilizzo in rete.

6.1.1 Formato

La classe dispone di una opzione globale internal che, se specificata, modifica completamente lo stile della lettera in uno essenziale e privo dei riferimenti aziendali. Il logo è riprodotto per mezzo del pacchetto grafico TikZ e le testatine ed i piè di pagina si adeguano automaticamente per le pagine successive alla prima.

6.1.2 Il tag di protocollo

A sinistra, subito al di sopra della linea che richiama l'oggetto della lettera, viene disegnato il riquadro del protocollo a seconda che il documento sia interno od esterno. Nelle aziende infatti esistono due protocolli distinti. Le due versioni si possono consultare nelle due porzioni di pagina riportate nella figura 1.

Una volta noti data e numero di protocollo, la lettera può essere ricompilata dopo aver inserito manualmente nel sorgente questi dati come argomento di due corrispondenti comandi. In questo modo la lettera diviene un archivio ufficiale, anche se solamente locale.

6.1.3 Gli indirizzi

Nella classe documento è stato definito un comando che compone gli indirizzi dei destinatari a destra del riquadro del protocollo. L'argomento può contenere semplici macro il cui nome richiama quello degli uffici interni e macro il cui nome corrisponde a cognomi di colleghi. Con questi *alias* mnemonici è assai rapido specificare destinatari con relativa qualifica o comporre le firme dei mittenti.

Se l'altezza dell'indirizzo aumenta oltre una limite stabilito — come quando vi sono più destinatari — lo spazio necessario viene ricavato automaticamente abbassando il riquadro del protocollo e la riga dove è riportato dell'oggetto.

6.1.4 Le firme

Nella classe sono definite due macro per comporre le firme in varie configurazioni: la prima stampa una firma sulla destra della pagine e la seconda ne stampa due affiancate. Entrambe le macro verificano che ci sia abbastanza spazio nella pagina per apporre fisicamente la firma e — con opzioni chiave/valore — stampano a richiesta le sigle sul lato sinistro e una riga di base.

6.1.5 Sviluppi futuri

Utilizzare questa classe risulta piuttosto semplice; le lettere sono sempre invariabilmente conformi allo standard aziendale ma la gestione dati può essere ulteriormente sviluppata. La procedura di protocollo, per esempio, può essere integrata con un database in rete sull'esempio dell'esperienza fatta dal Comune di Napoli e ampiamente illustrata nell'articolo De Marco e Cresci (2011).

I sorgenti delle lettere prodotti dagli utenti, anche nelle versioni intermedie da approvare, potrebbero essere memorizzati nel database centrale e bloccati in modifica una volta assegnato il codice di protocollo, per poi essere spediti in automatico ad indirizzi di e-mail PEC.

6.2 Un grafico

I grafici possono essere creati direttamente con gli strumenti di LaTeX. Il pacchetto più utile è pgfplots (De Marco e Giacomelli, 2011), che consente di mantenere la coerenza tipografica con il resto del documento e rifinire il grafico con gli elementi più vari.

L'esempio riguarda il grafico di *spettri elastici* relativi ad un sito di costruzione di un progetto reale: si tratta di curve matematicamente definite su quattro intervalli di tempo contigui, come si può osservare nella figura 2. Il codice del grafico è già stato dettagliatamente illustrato nel citato articolo, ma quello qui presentato è stato generato da una libreria in Lua, un linguaggio di scripting dai molti vantaggi, anziché direttamente dall'utente.

Questa prospettiva mostra come i sorgenti LaTeX, essendo semplici file di testo, possono essere generati automaticamente da programmi esterni senza dover far ricorso a particolari librerie. Ritorneremo sull'argomento e sulle modalità di generazione automatica degli spettri forse in futuri articoli, poiché si tratta di una tecnica utile.

Il codice riportato interamente di seguito mostra come il grafico sia generato come singolo documento nel formato PDF, già ritagliato nelle sue dimensioni minime di pagina grazie alle proprietà della classe standalone, e pronto per l'inserimento nel documento principale della relazione tecnica. La procedura evita che il codice del grafico debba essere generato ad ogni compilazione del documento principale, come succederebbe se lì fosse inserito, e consente all'utente di controllare più comodamente il risultato per gli affinamenti del caso.

```
% make with 'Spectrum' Lua library
% !TEX encoding = UTF-8
% !TEX program = pdflatex

\documentclass{standalone}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[italian]{babel}

\usepackage{lmodern}
\usepackage{pgfplots}
\usepackage[output-decimal-marker={,},
            arc-separator=\,]{siunitx}

\pgfplotsset{compat=1.5}

\newbox\panel
\setbox\panel=\hbox{\rule{2pt}{0pt}%
\scriptsize
\vbox{%
\hbox{Sito: Licciana Nardi (MS),
località Ponte Nuovo}
\hbox{Latidudine ED50
$\varphi=\ang{44;16;5.29}$}
\hbox{Longitudine ED50
      $\lambda=\ang{10;02;37.98}$}
\hbox{Categoria di suolo: E}
\hbox{Categoria topografica T1
    ($S_\mathrm{T}=\num{1.00}$)}}}

\newdimen\hg
\hg=7.2cm

\def\ymax{1.100}

\newdimen\yp
\yp=0pt
\advance\yp by \ht\panel
\advance\yp by \dp\panel
\advance\yp by 5pt% margine sup.

\pgfmathparse{\ymax*(1-\yp/\hg)}
\let\ylegend\pgfmathresult

\advance\yp by 5.6mm% spazio per legenda
\pgfmathparse{\ymax*(1-\yp/\hg)}
\let\ybox\pgfmathresult

% prima ascissa del box testuale da regolare
% eventualmente a mano:
\def\xbox{1.32}
\pgfmathparse{\xbox+0.025}
\let\xlegend\pgfmathresult

\begin{document}
\begin{tikzpicture}
\begin{axis}[
      legend style={draw=none,
        at={(axis cs:\xlegend, \ylegend)},
        anchor=north west},
      legend columns=4,
      tick label style={font=\scriptsize},
      xmin=0,
      ylabel={$S_\mathrm{e}/g$},
      ymin=0,
      grid=major,
      ymax=\ymax,
      scale only axis,
      xlabel={$T$ (s)},
      width=11.5cm,
      height=\hg,
      variable=\T,
      xtick={0.0,0.5,1.0,1.5,2.0,2.5,3.0},
      xmax=3.0]

% slc
\addplot[color=red,
         line width=1.4pt] coordinates {
         (0.000,  0.404)
         (0.179,  0.979)
         (0.536,  0.979)
};
\addplot[color=red,
         samples=150,
         domain=0.536:2.609,
         line width=1.4pt] {0.52433/T};
\addplot[color=red,
         samples=150,
         domain=2.609:3.000,
         line width=1.4pt] {1.36794/T^2};

% slv
\addplot[color=blue,
```

```
        line width=1.4pt] coordinates {
        (0.000,  0.314)
        (0.179,  0.758)
        (0.536,  0.758)
};
\addplot[color=blue,
        samples=150,
        domain=0.536:2.384,
        line width=1.4pt] {0.40639/T};
\addplot[color=blue,
        samples=150,
        domain=2.384:3.000,
        line width=1.4pt] {0.96881/T^2};

% sld
\addplot[color=orange,
        line width=1.4pt] coordinates {
        (0.000,  0.123)
        (0.167,  0.300)
        (0.501,  0.300)
};
\addplot[color=orange,
        samples=150,
        domain=0.501:1.907,
        line width=1.4pt] {0.15015/T};
\addplot[color=orange,
        samples=150,
        domain=1.907:3.000,
        line width=1.4pt] {0.28636/T^2};

% slo
\addplot[color=green,
        line width=1.4pt] coordinates {
        (0.000,  0.095)
        (0.163,  0.230)
        (0.488,  0.230)
};
\addplot[color=green,
        samples=150,
        domain=0.488:1.837,
        line width=1.4pt] {0.11223/T};
\addplot[color=green,
        samples=150,
        domain=1.837:3.000,
        line width=1.4pt] {0.20613/T^2};

% legenda spettri:
\legend{\textsc{slc},,,
        \textsc{slv},,,
        \textsc{sld},,,
        \textsc{slo},,,
}
% add text info
 \draw[black,fill=white]
   (axis cs:3.000,\ybox)
   rectangle
   (axis cs:\xbox,1.100)
   node[anchor=north west]
     {\box\panel};
\end{axis}
\end{tikzpicture}
\end{document}
```

6.3 Una relazione tecnica

Come per le lettere, anche per le relazioni tecniche sono state messe a punto delle apposite classi di documento che si occupano di definire i parametri tipografici della pagina, dei font ed i comandi per facilitare la stesura dei documenti. Una pagina di esempio è riportata nella figura 3.

6.3.1 I dati comuni

Una delle particolarità che è stata affrontata con semplici accorgimenti del sistema TEX tratta della gestione dei dati comuni a tutti i documenti di uno stesso progetto.

Si realizza il concetto di centralizzare questi dati per evitarne la ridondanza; una singola modifica a questi dati, come per esempio la modifica dell'indirizzo del committente, si riflette così automaticamente e senza errori in tutti i documenti (con l'avvertenza di ricordarsi di ricompilare i sorgenti).

L'implementazione è semplice ma efficace: un file di testo nominato con il numero di commessa e di estensione `.dat` viene posizionato nell'albero locale della distribuzione TEX — quindi visibile da ogni cartella dell'utente. In esso, i dati comuni sono associati a chiavi attraverso una macro chiamata `\set`. Un frammento di codice del file dati è riportato di seguito:

```
% lavoro 151
...
\set{progdate}{28/09/2012}
\set{comune}{Licciana Nardi}
\set{jobnumber}{151}
\set{lotto}{01}
\set{progfase}{E0}
...
```

La classe documento — che possiamo chiamare `tecrel`— implementa attraverso il pacchetto xkeyval l'opzione job a cui viene assegnato il numero di commessa, come nel seguente esempio:

```
\documentclass[job=151]{tecrel}
```

Durante la compilazione, il file `151.dat` viene caricato così da rendere disponibili attraverso le chiavi i dati salienti del corrispondente progetto, alle macro di composizione automatica del frontespizio, dei nominativi dei responsabili, dell'identificazione del documento, eccetera.

6.3.2 Sviluppi futuri

Come per le lettere aziendali illustrate nell'esempio della sezione 6.1, anche per le relazioni tecniche è possibile trasferire i dati salienti dei progetti in database di rete. Un archivio di rete di questo tipo potrebbe essere esteso per giungere alla completa gestione delle commesse.

Se invece è un software commerciale di rete a gestire le commesse all'interno dell'organizzazione, dovremo assicurarci che il gestionale in questione sia basato su un database server dotato dei driver di connessione per i più diffusi ambienti di sviluppo.

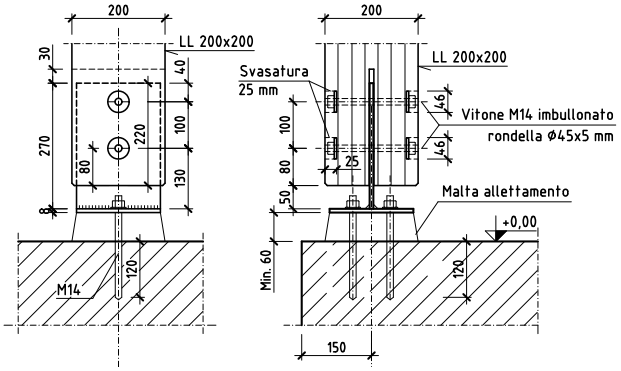


Figura 50 – Dettaglio del collegamento di base della colonna centrale.

dove λ è la massima snellezza della colonna che in effetti è vincolata con un asta in legno nella mezzeria sia in direzione del piano verticale longitudinale contenente il telaio, sia in direzione trasversale. Cautelativamente si assume come non efficace il vincolo di mezzeria pertanto la snellezza massima considera l'intera altezza dell'elemento supposto vincolato agli estremi con cerniere.

$$\lambda = \frac{L}{\rho} = \frac{545}{5{,}77} = 94{,}4. \tag{131}$$

Il raggio d'inerzia si è calcolato con:

$$\rho = \sqrt{\frac{J_{\min}}{A}} = \sqrt{\frac{13\,333}{20 \cdot 20}} = 5{,}77\,\text{cm}. \tag{132}$$

L'espressione per il calcolo del $k_{\text{crit,c}}$ data dalla norma è la seguente, essendo $\lambda_{\text{rel,c}} > 0{,}3$:

$$k_{\text{crit,c}} = \frac{1}{k + \sqrt{k^2 - \lambda^2_{\text{rel,c}}}} = \frac{1}{1{,}713 + \sqrt{1{,}713^2 - 1{,}518^2}} = 0{,}399. \tag{133}$$

con (il coefficiente $\beta_{\text{c}} = 0{,}1$ per il legno lamellare):

$$k = 0{,}5\left[1 + \beta_{\text{c}}\left(\lambda_{\text{rel,c}} - 0{,}3\right) + \lambda^2_{\text{rel,c}}\right] = 0{,}5\left[1 + 0{,}1\left(1{,}518 - 0{,}3\right) + 1{,}518^2\right] = 1{,}713. \tag{134}$$

12.3.3 Verifica collegamento di base colonne

La resistenza del collegamento di base delle colonne mostrato in figura 50, secondo le espressioni dell'Eurocodice 5 §8.2.3 è il minimo valore delle resistenza di tre meccanismi di rottura e snervamento della teoria di Johansen: il rifollamento del legno e due modi di rottura del bullone, valutabili con le seguenti espressioni:

$$F_{\text{v,Rk}} = min \begin{cases} f_{\text{h,1,k}} t_1 d; \\ f_{\text{h,1,k}} t_1 d \left(\sqrt{2 + \dfrac{4 M_{\text{y,Rk}}}{f_{\text{h,k}} d t_1^2}} - 1\right) + \dfrac{F_{\text{ax,Rk}}}{4}; \\ 2{,}3\sqrt{M_{\text{y,Rk}} f_{\text{h,k}} d} + \dfrac{F_{\text{ax,Rk}}}{4}. \end{cases} \tag{135}$$

FIGURA 3: Una pagina estratta da una relazione di calcolo strutturale di un progetto reale. Un particolare costruttivo è incluso in formato vettoriale perfettamente in scala e direttamente dal programma CAD, mentre alcune formule rappresentano le verifiche del giunto.

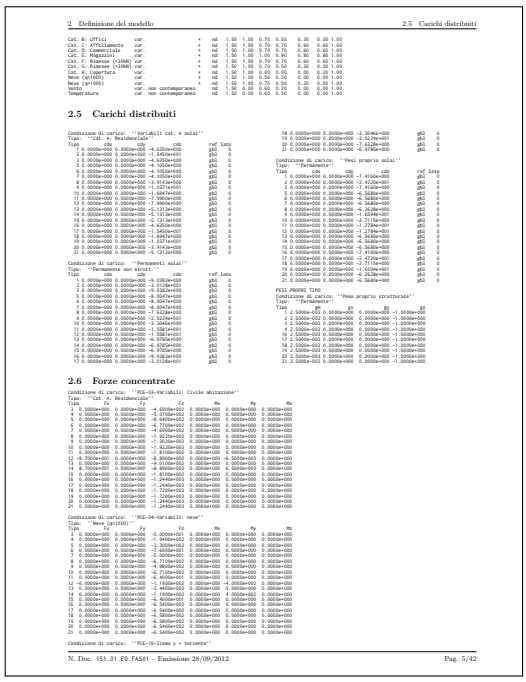


FIGURA 4: Una pagina estratta da un fascicolo di calcolo relativo all'analisi di un sistema di fondazione superficiale di un edificio. Per esso, la procedura illustrata nel testo per l'importazione dei dati da file `.rtf` ha prodotto 23 file di testo, ciscuno contenente l'insieme dei dati numerici relativi ad un singolo aspetto del modello (per esempio, l'elenco delle forze applicate di tipo distribito o concentrato). Nella stessa pagina una tabella numerica è composta a due colonne ed inframezzata da materiale impaginato a tutta larghezza.

6.4 Un fascicolo di calcolo

Un fascicolo di calcolo è un rapporto numerico formato da lunghe tabelle. In ciascuna di esse vi sono i dati che definiscono completamente un particolare aspetto del modello di calcolo agli elementi finiti utilizzato per le analisi strutturali. Per esempio, la prima tabella è solitamente l'elenco dei nodi a cui è associato un numero con le relative coordinate nel sistema di riferimento $O(x, y, z)$, mentre le ultime contengono i valori delle sollecitazioni che determinano la risposta alle azioni statiche e sismiche e le verifiche strutturali negli elementi.

Poiché la larghezza di queste tabelle varia molto una dall'altra, è conveniente regolare il numero di colonne per risparmiare spazio e contenere il numero delle pagine totali.

Il documento deve essere dotato del frontespizio conforme a quello degli altri documenti di progetto, di un sommario e delle opportune testatine e piè di pagina d'identificazione, come è possibile notare nella pagina di esempio di figura 4.

6.4.1 Importazione dei dati

La gestione delle colonne, nell'esempio di documento, è realizzata con le funzionalità del pacchetto multicol di Frank Mittelbach (l'attuale *leader* del LaTeX Project Team); composizione questa particolarmente onerosa per i programmi di word processing. Tuttavia, la maggior parte dei programmi di analisi agli elementi finiti del settore dell'ingegneria civile predispone i file di uscita in formati come il `.doc` o `.rtf`. Se così fosse, nascerebbe il problema di come importare i dati dei modelli ed i risultati nel formato testo.

La soluzione proposta è anch'essa piuttosto semplice e va eventualmente adattata allo specifico formato perché *artigianale*. I suoi passi sono riportati nel seguente elenco e si basano sull'allineamento naturale prodotto quando le righe hanno tutte lo stesso numero di caratteri e spazi con numeri ben incolonnati, ed il font è mono-spaziato:

1. una volta aperto il file con il word processor si esegue un copia-incolla di tutto il testo dall'oggetto tabella all'editor di testo;
2. si eliminano eventuali intestazioni iniziali e si salva un primo file;
3. all'interno del sorgente si include il file così costruito regolando il numero di colonne in base alla larghezza della riga;
4. si ripete la procedura per la successiva tabella.

Il comando di inserimento fa uso del pacchetto fancyvrb ed è il seguente (presuppone che i file siano salvati nella cartella locale `dati`):

```
\newcommand{\insertfile}[1]{%
\VerbatimInput[fontsize=\scriptsize,
     baselinestretch=0.86]{dati/#1}}
```

6.5 Un elenco documenti

Può essere obbligatorio allegare al progetto l'elenco dettagliato della documentazione. Con le classi personalizzate non è un problema definire un sorgente che genera il documento importando i dati da un file: conviene infatti separare il sorgente del documento dai dati creando un secondo file di testo in un comodo formato. Nel file dati dell'esempio si trova questo frammento di codice:

```
\row{151.01.E0.REL01}{Relazione tecnica}
\row{151.01.E0.REL02}{Relazione geotecnica}
\row{151.01.E0.REL03}{Relazione materiali}
\row{151.01.E0.REL04}{Relazione fondazioni}
\row{151.01.E0.REL05}{Relazione di calcolo}
```

Il comando `\row`, definito nel preambolo del sorgente principale, fa uso di un contatore per numerare automaticamente le righe, inserendo il codice identificativo del documento e la sua descrizione con la sintassi di una tabella standard di LaTeX a tre colonne:

```
\newcount\nr
\newcommand\row[2]{%
   \global\advance\nr by 1\relax
   \the\nr & \texttt{#1} & #2\\\midrule}
```

La tabella dell'elenco dei documenti di progetto può eccedere la singola pagina perciò, per risolvere il problema, si è utilizzato il pacchetto longtable.

Nel corpo del sorgente principale il codice seguente definisce un ambiente longtable nel quale sarà riversato il contenuto del file chiamato DATI.tex con l'intero elenco:

```
\begin{longtable}{p{7mm}p{32mm}|p{105mm}}
N.&   Identificativo & Contenuto documento\\
\midrule
\endfirsthead
N.&   Identificativo & Contenuto documento\\
\midrule
\endhead
\multicolumn{3}{r}{segue \dots}\endfoot
\multicolumn{3}{r}{fine lista.}\endlastfoot
\input{DATI}
\end{longtable}
```

7 Conclusioni

La documentazione di progetto non è essenzialmente diversa dal tipo di testo che ispirò Donald Knuth nel creare il sistema TEX. Nel tempo si sono aggiunti strumenti come i programmi bibliografici ed i formati ad alto livello come LATEX dando all'utente una capacità molto maggiore di produrre documenti, in particolare quelli dei progetti di ingegneria. Le potenzialità del sistema sono tuttora in evoluzione e già oggi è in grado di soddisfare le esigenze compositive più complesse, in un ambiente di lavoro che favorisca ulteriormente la qualità della documentazione prodotta.

Chiaramente, in ogni ambito professionale le persone possono giudicare più o meno adatto uno strumento di lavoro in funzione delle preferenze, delle abitudini, delle esigenze pratiche e del desiderio di qualità.

Il vantaggio dell'utilizzare un sistema di composizione tipografica basato su un linguaggio consiste nel risolvere i problemi con grande flessibilità e con poche limitazioni. Lo dimostra il sistema TEX, che soddisfa brillantemente i requisiti insiti nella documentazione di progetto.

Per iniziare, occorre un buon impiego di risorse per impadronirsi dei concetti di base, scrivere macro od imparare l'uso dei pacchetti già disponibili, abituandosi alla sintassi di un linguaggio piuttosto che utilizzare il mouse con pulsanti e dialoghi d'interfaccia. In poche parole potremmo riassumere confermando che *l'impegno consapevole è ripagato da ottimi risultati*.

8 Ringraziamenti

L'idea su cui si basa questo articolo è quella di far presente ai tecnici autori di documentazione come il sistema TEX proponga strumenti validi per risultati e affidabilità, da prendere in seria considerazione per la propria attività.

Pertanto ringrazio i colleghi che in passato mi hanno chiesto di illustrare le caratteristiche dei motori di composizione e pacchetti e gli studenti ed i professionisti che vorranno valutarlo con attenzione.

Ringrazio come al solito la mia famiglia per il sostegno ed il tempo che mi ha concesso per questo lavoro, ma stavolta non è questo l'ultimo riconoscimento: un particolare grazie va infatti a tutte le persone che hanno lavorato e collaborato con passione per realizzare questo numero di ArsTEXnica ed organizzare il GUIT *meeting 2013*.

Riferimenti bibliografici

Beccari, C. (2009). «Il formato archiviabile dei file PDF». *ArsTEXnica*, (7), pp. 13–24. URL http://www.guitex.org/home/it/arstexnica.

— (2011). «XeLATEX and the PDF archivable format». *ArsTEXnica*, (12), pp. 6–11. URL http://www.guitex.org/home/it/arstexnica.

— (2013). «Introduzione alla creazione di file di classe». http://www.guitex.org/home/images/doc/GuideGuIT/introcreaclassi.pdf.

Beccari, C., Canavero, F., Rossetti, U. e Valabrega, P. (2013). «Saper comunicare, cenni di scrittura tecnico-scientifica». http://www.guitex.org/home/images/doc/sapercomunicare.pdf.

De Marco, A. e Cresci, P. E. (2011). «LATEX nella pubblica amministrazione. La web application FACILE per la produzione automatica di comunicazioni interne del Comune di Napoli». *ArsTEXnica*, (12), pp. 39–56. URL http://www.guitex.org/home/it/arstexnica.

De Marco, A. e Giacomelli, R. (2011). «Creare grafici con pgfplots». *ArsTEXnica*, (12), pp. 12–38. URL http://www.guitex.org/home/it/arstexnica.

Erika Leonardi, a. c. d. (2013). «Comunicare: impegno a farsi capire». *Unificazione & Certificazione*, **3**, p. 25.

Myers, J. D. (2013). «Assemblaggi virtuali». *Le Scienze*, (540), pp. 94–95.

Rothenberg, J. (1995). «La conservazione dei documenti digitali». *Le Scienze*, **LIV** (319), pp. 16–21.

Scarso, L. (2010). «PDF/A-1a in ConTEXt». *ArsTEXnica*, (10), pp. 25–32. URL http://www.guitex.org/home/it/arstexnica.

▷ Roberto Giacomelli
Carrara (MS)
giaconet dot mailbox at gmail dot com

Gli alfabeti romani di Francesco Griffo

Claudio Vincoletto

Sommario

Presentiamo un timido e sperimentale approccio per riprodurre alcuni caratteri aldini con METAFONT, tenendo presente la natura calligrafica che soggiace alle prime incisioni tipografiche ed i limiti del tratto con penne fisse.

Abstract

We show a rudimental approach to Griffo types with METAFONT. Fixed pens and architectural criteria are used, to reproduce the ancient flavour of the aldine press.

1 Premesse

Diversi alfabeti romani furono commissionati al bolognese Francesco Griffo da Aldo Manuzio per la sua celebre officina. Oggi questi caratteri rappresentano una pietra miliare per la tipografia e possono costituire ancora un valido metro di confronto per chiunque voglia dare vita ad un carattere limpido e leggibile.

Le ragioni vanno ricercate nel successo che ebbero da subito questi alfabeti, riconosciuti come un modello da seguire per molti maestri tipografi della scuola francese in epoca rinascimentale, fino ai revival del Novecento.

La collaborazione di Griffo con Manuzio durò per almeno otto anni, fino al 1501. Malgrado diverse attestazioni gli attribuiscano il mestiere di scultore, orafo e stampatore, molti studiosi propendono a ritenere fondamentale la sua formazione di calligrafo, proprio per l'evidente padronanza delle varietà e delle forme di scrittura. A lui viene riconosciuta la prima incisione di un carattere corsivo, oltre che un'originalità senza eguali nel dare forma a caratteri tondi, sempre equilibrati ed eleganti.

Il più celebre è sicuramente quello impiegato per la prima volta nell'edizione del *De Aetna* di Pietro Bembo (tondo 4),[1] successivamente adottato, con una diversa serie di maiuscole, anche nell'*Hypnerotomachia Poliphili*. Così lo presenta Giovanni Mardersteig:

> Il carattere Bembo è il capolavoro di Francesco da Bologna ed unisce in modo esemplare l'eleganza con la severità della forma. Esso rappresenta la perfetta trasposizione di una nobile *littera humanistica* in carattere da stampa. E tuttavia sono talmente numerosi in questo carattere gli indizi della mano di un calligrafo, che proprio qui può bene trovare appoggio nell'esemplificazione che Griffo sia stato originariamente uno *scriptor* [Mardersteig (1988), 135].

Nel presente articolo vengono illustrate una serie di strategie per sviluppare una famiglia di font, denominata provvisoriamente *Computer Griffo*, che mira a riprodurre il famoso alfabeto utilizzato da Aldo. I criteri messi a punto tengono presente l'armonia strutturale approntata da Griffo, la sua intrinseca natura calligrafica, i limiti e le possibilità di METAFONT.

2 I revival del Novecento

Malgrado la profonda influenza esercitata dal disegno di Griffo nei secoli successivi (ne furono soggetti De Colines, Estienne, Garamont, Granjon, tra i maggiori), con l'avvento dei tipi moderni, ad asse verticale, i caratteri di derivazione aldina caddero nell'oblio. Il nome di Griffo ricompare in alcuni scritti specialistici, ad esempio nell'introduzione del Manuale Tipografico di Bodoni, però senza che ne vengano conservati i tratti e le peculiarità.

Nel XX secolo invece, sull'onda del recupero dei caratteri tipici degli incunaboli veneziani, spetta a Stanley Morison la riscoperta dell'importanza di Griffo. Sua la supervisione alla Monotype per l'incisione dapprima del Polifilo e poi del Bembo. Nonostante il successo commerciale del carattere Bembo (uno dei font più utilizzati nell'editoria del Novecento), Morison non era soddisfatto del disegno finale, troppo rispettoso dei limiti imposti dalle macchine Monotype, e propose all'amico Mardersteig, coadiuvato dall'incisore Charles Malin, di ritentare la riproduzione del carattere del *De Aetna*, in modo più filologico. Il risultato fu il carattere Griffo, disponibile solo per la stampa a torchio dell'Officina Bodoni.

Del resto successivamente anche il Dante, sempre di Malin/Mardersteig, deve moltissimo all'originale di Griffo, come d'altronde una quantità enorme di font digitali ben conosciuti, tra cui il Minion di Robert Slimbach o il più recente Neacademia di Sergei Egorov.

1. Una discreta digitalizzazione, di questo ed altri incunaboli citati, può essere consultata qui: `www.digitale-sammlungen.de`.

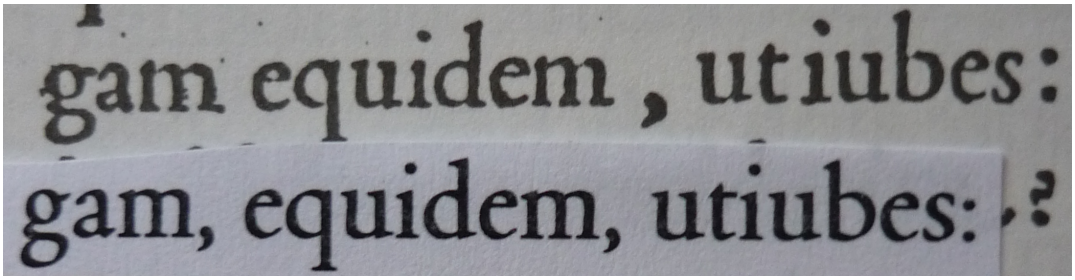


Figura 1: Computer Griffo sul modello di Mardersteig, stampa laser a paragone con il carattere del *De Aetna*.

3 Alcune considerazioni paleotipografiche

Una traduzione in digitale sufficientemente fedele dei caratteri di Griffo presenta enormi difficoltà. Le stampe sopravvissute, sebbene di qualità, presentano una serie di alterazioni accidentali, increspatura e restringimento della carta, caratteri fusi in modo approssimativo, sbavature di inchiostro, artefatti di varia natura, da rendere vano ogni tentativo univoco di riproduzione.

La scelta che abbiamo operato si rifà quindi ad un ragionamento di diverso ordine. Ai tempi di Griffo, la stampa non godeva di buona reputazione. I libri impressi venivano giudicati di qualità inferiore rispetto alla produzione manoscritta, pertanto gli incisori/punzonatori di caratteri si sforzavano di mimare la scrittura a penna nei minimi particolari. Sicuramente questo è il caso del Griffo, e i tipi del *De Aetna* ne sono un esempio lampante. Ogni glifo è appositamente fornito di più varianti, con svolazzi o meno, così da rompere la monotona regolarità della riga stampata, simulando la vivezza della scrittura calligrafica. Probabilmente il fascino esercitato dal carattere di Griffo dipende anche da questo fattore.

Un'ulteriore componente, ritenuta cruciale, è la disposizione armonica dei tratti, che nelle nostre analisi celano spesso una struttura geometrica. Anche questo aspetto non deve stupire, visto che in epoca rinascimentale sono numerosi i trattati che forniscono istruzioni per la costruzione di alfabeti secondo dettami matematici: Feliciano, Pacioli, Mercator, tra i più famosi.

A partire da questi presupposti, METAFONT ci è sembrato lo strumento ideale per creare un archetipo di font in grado di rispettare tali requisiti. L'organizzazione del progetto ha richiesto un approccio diverso rispetto al disegno del Computer Modern. Disponiamo essenzialmente di tre modi per operare con plain METAFONT. Per mezzo dell'istruzione `fill`, delineiamo semplicemente dei contorni che verranno campiti; con `draw`, si stendono dei tratti con penne predefinite nelle dimensioni, che però non possono variare l'angolatura; con `penstroke`, si opera in una modalità intermedia, per cui ad ogni punto cardine possono essere assegnate una larghezza ed un'inclinazione, ma la macro restituisce un tratto generato con una penna irrimediabilmente piatta. Knuth ha inoltre sviluppato per il Computer Modern una serie di macro, (`filldraw stroke`) atte a simulare un pennino a punta dilatabile (*pointed nib*) tipico nei caratteri moderni, in grado di creare dei contorni con penne circolari minute, denominate penne virtuali. Nel nostro caso, vista la problematicità di conservare l'andamento di una penna fissa (*broad nib*), soprattutto nei tratti curvi, abbiamo fatto uso delle potenzialità intrinsecamente calligrafiche del programma (specificatamente tramite il comando `draw`), sebbene limitate. Di questo discuteremo in seguito.

In ogni caso, abbiamo tenuto presente i rari font che nel passato sono stati programmati con questa tecnica. In particolare:

- le *Calligraphic Capitals* (`calu.mf`) di Billawala, presenti nella collezione del Computer Modern;
- il *Vicentino* di Kraml, disponibile su CTAN;
- *AlQalam* di Sherif e Fahmy (alcuni articoli su TUGboat).

4 Misure e proporzioni

In linea generale, per quel che concerne la struttura e la dimensione dei caratteri, abbiamo seguito le indicazioni fornite da Peter Burnhill, nel suo studio dedicato alle matrici utilizzate nell'officina di Manuzio [Burnhill (2004)]. A partire da un'accurata serie di misurazioni, comprendenti anche una copia del *De Aetna*, egli giunge alla conclusione che probabilmente Manuzio abbia adottato una specifica unità di misura, a partire dal carattere utilizzato nella sua edizione di Costantino Lascaris, che viene mantenuta nel successivo impiego di corpi minori. L'unità equivale alla dodicesima parte della matrice (la suddivisione segue l'esempio di Mercator), determinata in questo caso dall'avanzamento di riga, che all'epoca corrispondeva alle reali dimensioni del carattere, racchiuse tra le estremità delimitate dagli ascendenti e dai discendenti. L'intervallo ottenuto corrisponderebbe al più piccolo spazio distribuito orizzontalmente (*hair space*), e ciò sarebbe confermato dalla presenza di macchie

di inchiostro che ne rivelano l'impronta. Il valore dell'ipotetico spazio tipografico aldino corrisponde a 0.527 mm, e replicato undici volte, nel caso del *De Aetna*, il cui romano è più piccolo del *Lascaris* (tondo 1), corrisponde approssivamente a 5.8 mm.[2] Il valore non si discosta di molto dai dati rilevati con il tradizionale metodo Proctor-Haebler sulle venti righe, usato generalmente per lo studio degli incunaboli. Le precedenti misurazioni attribuiscono al carattere di Griffo un valore di 114 mm (ogni riga quindi di 5.7 mm). Burnhill vi aggiunge una percentuale minima, per compensare le deformazioni della carta di cui si è detto.

Nel *De Aetna* la dimensione del carattere suddivisa in dodicesimi presenta alcune particolarità. L'altezza della x si approssima a 5/12 del corpo, mentre gli ascendenti e i discendenti a 3.5/12. Per mantenere in METAFONT queste proporzioni, abbiamo tralasciato il criterio utilizzato da Knuth nel Computer Modern, ovvero quello di attribuire ad ogni parametro un valore prestabilito, lasciando calcolare al programma i rispettivi rapporti. L'esempio seguito è quello del *Vicentino*.

```
font_identifier:="cgrf"; font_size 5.8mm#;

size#:=5.8mm#;
hair_space#= 1/12 size#;
letter_fit#:=0mm#;

asc_height# + desc_depth# = size#;
x_height#= 5/12 size#;
asc_height#= x_height# + 3.5/12 size#;
cap_height#= 7.5/12 size#;
bar_height#= 3.5/12 size#;
```

L'analisi delle immagini riproducenti il carattere originale (alcuni fac-simile e il Griffo di Marbersteig) è stata effettuata con Inkscape, sovrapponendo uno schema di quadrettatura — procedimento che si è rivelato più flessibile e dinamico rispetto al suo impiego con GIMP. Inkscape è un software pensato per grafici vettoriali, eppure consente di lavorare anche con immagini raster, disegnando su più livelli. Successivamente, le immagini elaborate con Inkscape sono state importate in Geogebra, programma visuale che facilita lo studio della geometria, dell'algebra e dell'analisi. Anche in questo caso è possibile lavorare su più livelli, la figura raster come sfondo, i costrutti geometrici in primo piano.

2. Buona parte della ricerca di Burnhill è condotta a partire da ingrandimenti di fotocopie (almeno 8×), effettuate su fonti primarie, sulle quali è stata costruita una griglia di riferimento, basata sull'avanzamento di riga. La misurazione diretta dei vari caratteri successivi al *Lascaris* confermerebbe la tesi secondo cui tutte le matrici dell'officina aldina sarebbero derivate da questo prototipo: De Aetna (0.527×11), Herodotus (0.527×8), Octavo classics (0.527×7.5) [Burnhill (2004),70].

5 Definizione della penna e del tratto

Pur disponendo con METAFONT di un sofisticato strumento per creare delle penne in una qualsiasi forma desiderata, abbiamo provvisoriamente utilizzato delle penne standard per simulare una punta piatta. Come insegna Noordzij [Noordzij (2006)], ogni penna è sempre dotata di un certo spessore, che va ad incidere nello sviluppo del tratto, a seconda dell'angolazione. Abbiamo pertanto usufruito principalmente di una penna rettangolare per i tratti verticali e orizzontali, circolare per le curve (ma solo per ammorbidire l'asprezza generata dal *ductus* della penna rigida). Nel modulare i percorsi circolari, la penna rettangolare genera delle curve ad ogiva troppo acute, che esasperano le forme calligrafiche del glifo. Abbiamo quindi temporaneamente optato per un fluire più morbido, che risponde meglio alle impressioni dello stampato originale.

Il valore assegnato alla larghezza della penna equivale al già noto *hair space*. Questa nozione non è di Burnhill, ma deriva dalla tradizione calligrafica. All'epoca di Griffo si soleva infatti scrivere con un'altezza della x pari a 5-6 volte le dimensioni della penna, con un'angolazione di circa 25 gradi (in un primo momento si è sperimentato anche quello di 30 gradi). In alcune occasioni ci è sembrato opportuno individuare i punti estremi del rettangolo rappresentato dalla penna, per cui sono stati calcolati anche la diagonale e l'angolo da essa formato (arcotangente).

```
% pen
px#:= 1/5 x_height#; py#:= 1/4 px#;
pa:=25;
pa_o:=10;
diag#:=px#++py#;
diag_a:=pa-angle(px#,py#);
```

Questa tecnica ci agevola molto nel definire diversi corpi del carattere, poiché richiede l'intervento su pochissimi parametri. Ad esempio per creare un carattere simile al *Leoniceno* (tondo 5), usato per la prima volta da Manuzio nel 1497 per un'edizione del *Libellus de epidemia* ovvero *De morbo gallico*, è sufficiente allargare di poco l'occhio del carattere, riducendo gli ascendenti e i discendenti. In questo modo verrà automaticamente determinata la larghezza della penna e quindi generato un tratto più scuro, come si addice ai font più piccoli, per mantenere visivamente un più equilibrato colore della pagina. Nel nostro esperimento il Leoniceno viene configurato a partire dai 4.2mm e trova una sufficiente corrispondenza con l'originale aldino. L'insieme permane equilibrato, anche se necessita di ulteriori e più mirate correzioni.

Per sviluppare il font sono comunque necessarie diverse penne fisse. Questa è la definizione delle due principali:

```
pickup pensquare xscaled px yscaled py
     rotated pa;
stem_nib:=savepen;
pickup pencircle xscaled 1.1px yscaled py
     rotated pa;
bowl_nib:=savepen;
```

Come si può notare, la penna che descrive le curve non è solo senza spigoli, ma presenta un'ampiezza maggiore. Questa scelta è stata adottata per almeno due ragioni. Innanzitutto, perché i punti estremi di una penna inclinata costruita su un'ellisse, quando definiscono un'asta verticale, producono una proiezione più piccola sulla linea di base, rispetto ad una penna rettangolare. Inoltre, le linee curve risultano otticamente più minute delle aste, quindi questo valore va compensato.

Un grande limite nell'uso delle penne fisse con METAFONT è dovuto all'impossibilità di modificare l'angolatura della penna nel corso del tratto. Per ovviare a questo inconveniente, abbiamo attinto dall'esperienza di Sherif, maturata in una ricerca volta alla costruzione di glifi caratteristici della calligrafia araba. Sherif descrive una serie di macro, che vengono appositamente studiate per colmare tale lacuna. Al fine di testarne il funzionamento, visto che le macro non sono state presentate integralmente, è stato necessario operare in modo retroingegneristico.

Riassumendo, Sherif esplora la possibilità di associare ad ogni passo del tracciato di una penna $(z_{k+1} - z_k)$ un corrispettivo angolo di inclinazione, così che in ogni punto l'angolazione della penna sia un'interpolazione tra l'angolo di partenza e quello di arrivo. Lo scorrere di un finito numero di punti, come una scansione di singole impronte della penna (`drawdot`), dà l'effetto di un tratto continuo. Per approfondire l'argomento, rimandiamo ai riferimenti indicati in bibliografia [Sherif (2008)].

Nel nostro caso purtroppo i vantaggi sono circoscritti. Solo in un caso (nella lettera *e*), l'impiego delle macro si è rivelato produttivo; all'opera, forse perché non ben congegnato, il procedimento risulta piuttosto fragile. Sicuramente l'argomento è da approfondire, come anche l'intrinseca possibilità di perfezionare le macro.

Per ciò che concerne le metriche, adoperando il valore dell'unità fondamentale (1/12 dell'altezza) anche per la componente orizzontale dei glifi, abbiamo riscontrato una particolare corrispondenza con quanto illustrato da Frank Blokland in merito alla spaziatura fra le lettere [Blokland (2013)]. Egli ha elaborato una teoria che ritiene plausibile annoverare agli incisori di punzoni rinascimentali un comune criterio di costruzione dei glifi, secondo cui l'insieme delle proporzioni è ricondotto a modelli geometrici. Il discorso vale non solo per le capitali romane, ma anche per il minuscolo calligrafico umanistico, filtrato da un'idea di perfezione unitaria. L'uso delle griglie sta alla base di questo sistema, definito armonico, che ingloba, attraverso il modello che viene a costituire, sia un sistema proporzionale che un sistema ritmico. La successione di intervalli regolari, le unità, non sono altro che frazioni della griglia, e coinvolgono ad un tempo sia la struttura del glifo che gli spazi interparola. In tal caso le componenti armonico-ritmiche coincidono con altri metodi comunemente adottati, come ad esempio la spaziatura determinata otticamente. La microunità, che abbiamo impiegato per definire gli spazi tra le lettere, è per il Computer Griffo di 1/72 del corpo; lo schema riprodotto è molto simile a quello esposto da Blokland, ma talvolta adattato qualora vengano ad emergere differenti configurazioni del contorno.

6 Costruzione dei glifi

Per esplicitare meglio come si è inteso sviluppare il Computer Griffo, forniamo un esempio concreto: la costruzione della lettera *d* minuscola. Scomponendo gli elementi del glifo, potremmo considerare tre componenti fondamentali: il tratto verticale (ascendente), il tratto curvo (altezza della *x*), le grazie (superiore e inferiore). In apertura si dànno le coordinate verticali ed orizzontali:

```
cgrfchar "The letter d";
beginchar("d",6hair_space#,asc_height#,0);
adjust_fit(0,0);
```

Il tratto verticale non presenta particolari problemi, poiché i punti cardine che lo delimitano sono facili da individuare.

```
pickup stem_nib;
rt x1=rt x2=w-hair_space;
top y1=h; bot y2=0;
draw z1..z2;
```

Diverso è il caso del tratto curvo. Con l'analisi effettuata con Geogebra, risulta chiaramente di forma ellittica, ma leggermente inclinata. Con METAFONT abbiamo bisogno di disporre di punti estremi per definire l'ellisse, e questi toccano la linea di base, l'altezza della *x*, la congiunzione con il tratto verticale, la distanza dal margine esterno sinistro. Sono imposte alcune correzioni ottiche: l'uso di *overshoots*, l'intersezione degli assi dell'ellisse a .52 dell'altezza della *x*. Per simulare l'inclinazione dell'ellisse, abbiamo impiegato un parametro esterno, il cui valore viene aggiunto o sottratto nei punti cardine:

```
pickup bowl_nib;
x5-bdpq_var=x7+bdpq_var=.5[x6,x0];
 x0=rt x1; lft x6=1/3hair_space;
top y5=x_height+o; bot y7=0-o;
 y0-bdpq_var=y6+bdpq_var=.52[y7,y5];
path p; p=superellipse(z0,z5,z6,z7,superness);
z4=(z1..z2) intersectionpoint p;
z8=(lft z1..lft z2) intersectionpoint reverse p;

path q; q=subpath(1,6) of p..z8;
draw q;
```

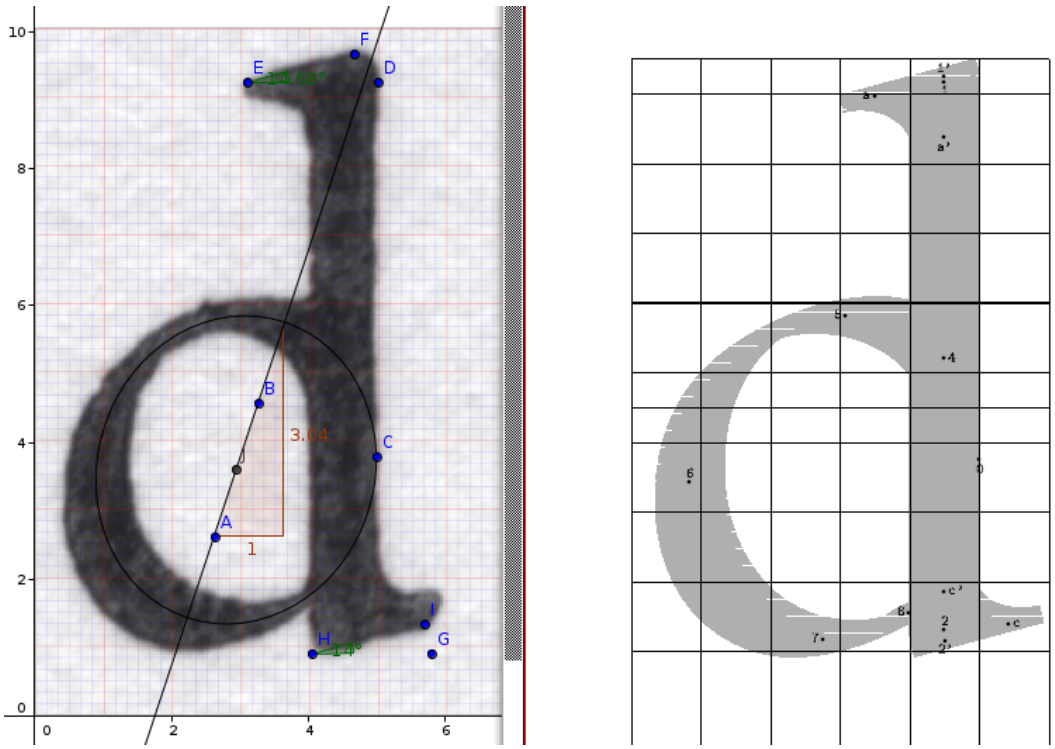


Figura 2: Relazione fra l'analisi con Geogebra e la lettera *d* generata con METAFONT.

La macro utilizzata per la grazia superiore cerca di mimare la mano del calligrafo. Viene definita l'inclinazione della penna, che estende il suo tracciato dal punto più alto, il movimento retto verso sinistra, quindi il movimento curvo per ricongiungersi all'asta verticale più in basso. La flessibilità di questo sistema ci permette di personalizzare al massimo ogni grazia, in inclinazione e lunghezza, ma sempre tenendo presente la struttura della penna ed il suo coerente movimento nello spazio piano.

```
vardef sloped_serif_top(suffix $,@)
  (expr phi,jut,bracket,height) =
% undraw z$--(x$,y$+py);
 pickup pensquare xscaled px yscaled py
    rotated phi;
 rt x$'=rt x$; top y$'=height;
 x@=x$'-jut*px; z$'-z@=whatever*dir phi;
 lft x@'=lft x$; bracket*(x@'-x@)=y@-y@';
 filldraw z$'--z@{dir phi}...{down}z@'--cycle;
 labels($',@,@') enddef;
```

Per la grazia inferiore abbiamo impiegato una macro simile, ma che incorpora la possibilità di uno spostamento in orizzontale della penna. Le istruzioni per la lettera *d* vengono concluse quindi in questo modo:

```
sloped_serif_top(1,a,serif_a-10,1,.6,h);
sloped_serif_bot(2,c,serif_a-10,.9,.5,
   1/hair_space);
penlabels(0,1,2,3,4,5,6,7,8); endchar;
```

Presentiamo un ulteriore esempio per esteso, la lettera *s*, mettendo in evidenza le problematiche di una penna che muta angolatura, così come sono state risolte senza adottare le macro di Sherif.

```
cgrfchar "The letter s";
beginchar("s",4hair_space#,x_height#,0);
adjust_fit(0,0);

pickup bowl_nib;
x0b-bdpq_var=x0d+bdpq_var=.5[x0a,x0c];
     x0c=0; x0a=w;
top y0b=h+o; bot y0d=0-o; y0c+bdpq_var=
     y0a-bdpq_var=.52[y0d,y0b];
path p; p=superellipse(z0a,z0b,z0c,z0d,superness);

z1=point 1 of p; z2=z0b;
z4=.5[z0c,z0a];
z6=z0d; z7=point -3 of p;
draw z2{left}..z4{dir -pa}..tension .8..{left}z6;

cutoff(z2,pa-90); cutoff(z6,pa+90);
pickup vertical_nib; drawdot z1; drawdot z7;

penpos1(diag,90-diag_a); penpos2(1.1px,pa);
fill z2r{right}..z1r--z1l..{dir 200}z2l--cycle;
penpos7(diag,90-diag_a); penpos6(1.1px,pa);
fill z6l{left}..z7l--z7r..{dir 10}z6r--cycle;

penlabels(0a,0b,0c,0d,1,2,2',3,4,5,6,7); endchar;
```

Il glifo può essere inscritto in un'ellisse lievemente inclinata verso destra. Su di essa giacciono anche i punti cardine, pertanto la prima operazione è proprio quella di definire l'ellisse nonché lo spazio vuoto ai margini. La curva della *s* passa dunque per il punto più alto `z2`, per l'incrocio degli assi dell'ellisse `z4`, infine per il punto più basso `z6`.

Rimangono da definire le grazie terminali, nei punti `z1` e `z7`. Si tracciano dei punti senza tratto (`drawdot`), ma con la penna disposta verticalmente. Quindi vengono campiti i tratti `z1..z2` e `z6..z7`, semplicemente delineandone i contorni con `fill`.

Non fornendo alcuna illustrazione del glifo, vogliamo far intendere come con METAFONT sia sempre necessario lavorare su un piano di astrazione molto elevato, nella previsione di un risultato che può essere testato solo in un secondo momento, quello della compilazione.

7 Conclusioni

Il Computer Griffo è attualmente solo ad uno stadio di abbozzo, eppure dimostra le potenzialità di METAFONT in percorsi non ancora esplorati.

Abbiamo appurato che la completa riproduzione dei glifi, nei loro minuziosi particolari, non può essere effettuata solo con l'uso di penne fisse, specie per effetti calligrafici tali da simulare un movimento del polso, uno scarto della penna, una minuscola rotazione dell'asse. In questi casi, siamo ricorsi all'istruzione `fill`, indicando il contorno voluto, anche se il ripercorrere la strada tracciata da Sherif ci sembra la scelta migliore.

Un ulteriore stadio di perfezionamento, oltre al completamento dell'alfabeto romano e della sua estensione ad altri alfabeti, prevede lo sviluppo di una penna personalizzata. Knuth le definisce *penne poligonali*. Si spera così di dare vita a quel disegno calligrafico ideale che Griffo aveva in mente, quando si accinse ad incidere i punzoni utilizzati nel *De Aetna*.

Riferimenti bibliografici

Blokland, F. (2013). «Harmonics, patterns, and dynamics in formal typographic representations of the latin script». URL `http://www.lettermodel.org`.

Burnhill, P. (2004). *Type spaces: in-house norms in the typography of Aldus Manutius*. Hyphen.

Carter, H., Carter, H. e Mosley, J. (2002). *A view of early typography up to about 1600*. Hyphen.

Johnston, E. (1906). *Writing & illuminating & lettering*. John Hogg. URL `http://archive.org/details/writingilluminat00johnuoft`. L'edizione presente online è una ristampa del 1917.

Knuth, D. (1986a). *Computer modern typefaces*. Computers & typesetting. Addison-Wesley.

— (1986b). *The METAFONTbook*. Computers & typesetting. Addison-Wesley.

Kraml, W. (2013). «Vicentino font». URL `http://www.ctan.org/tex-archive/fonts/vicentino/kraml`.

Mardersteig, G. (1988). *Scritti di Giovanni Mardersteig sulla storia dei caratteri e della tipografia*. Il Polifilo.

Morison, S. (1973). *A Tally of Types*. Cambridge University Press.

Noordzij, G. (2000). *Letterletter*. Hartley & Marks.

— (2006). *The Stroke: Theory of Writing*. Princeton Architectural Press.

Sherif, A. (2008). «Parameterized arabic font development for computer typesetting system». URL `http://eece.cu.edu.eg/~hfahmy/thesis/2008_01_paf.pdf`.

Sherif, A. e Fahmy, H. (2007). «Parameterized arabic font development for alqalam». URL `http://www.tug.org/TUGboat/tb29-1/tb91sherif.pdf`.

Smeijers, F. (1996). *Counterpunch: making type in the sixteenth century, designing typefaces now*. Hyphen.

Vervliet, H. (2008). *The Palaeotypography of the French Renaissance: Selected Papers on Sixteenth-century Typefaces*. Numero v. 1 in Library of the Written Word. Brill Academic Pub.

▷ Claudio Vincoletto
Torino
`claudio dot vincoletto at gmail dot com`

Il pacchetto minerva*

Grazia Messineo, Salvatore Vassallo

Sommario

Vengono illustrati gli sviluppi del pacchetto minerva, già introdotto al convegno GUIT 2012, per la somministrazione di esercitazioni e prove di valutazione al PC.

Abstract

We present the developments of the package minerva, already shown in 2012 GUIT conference, to produce exercises and exams which can be given by using a PC.

1 Introduzione

Il pacchetto minerva, che abbiamo introdotto durante il convegno GUIT 2012, consente di creare prove (esercitazioni o compiti) da somministrare agli studenti al PC.

Ricordiamo brevemente le principali caratteristiche illustrate in quell'occasione:

- le prove vengono create a partire da un database di esercizi, che i docenti hanno a disposizione e possono integrare;
- si deve compilare un file master, dal quale vengono generati tutti gli output necessari per l'esecuzione della prova (file interattivo, file cartaceo, stringa delle soluzioni);
- sono definiti vari comandi matematici di semplificazione delle frazioni, degli esponenti, ecc. (già condivisi con il pacchetto esami);
- è possibile inserire parametri casuali negli esercizi, in modo da generare dallo stesso file più prove "diverse";
- sono definite le opzioni compito ed esercitazione, che permettono di scegliere se si desidera costruire una prova di verifica vera e propria oppure un'esercitazione, che può essere tracciata oppure da svolgere offline;
- è previsto un ambiente solution anche per i test a risposta multipla;
- è previsto l'ambiente MNdb, che ha un solo argomento obbligatorio, ovvero il nome di un file nel quale saranno scritti gli esercizi e all'interno del quale il comando `\selectrandomlyn` permette di selezionare n varianti da un file e di scriverle su un file esterno. La chiusura dell'ambiente fa scrivere, in ordine casuale o fissato, gli esercizi selezionati. Il codice

```
\begin{MNdb}{equazioni}
\selectrandomlyn{3}{equadue}
\selectrandomlyn{6}{disequadue}
\end{MNdb}
```

  sceglie casualmente 3 esercizi dal file `equadue`, 6 dal file `disequadue` e li manda in output. I meccanismi di scelta casuale e riordinamento da una lista sono illustrati in dettaglio nel paragrafo 3.3.

2 acrotex

Il pacchetto minerva è costruito "sopra" i pacchetti di acrotex di D.P. Story.

La nostra idea è stata quella di estendere alcune funzionalità dei pacchetti preesistenti evitando il più possibile di rinunciare a quelle esistenti e automatizzare alcune procedure.

Con il nome acrotex ci si riferisce ad un gruppo di pacchetti, elaborati di D.P. Story per la generazione di esercizi ed esami. Uno degli aspetti caratteristici di acrotex è quello di sfruttare appieno alcune potenzialità dei file PDF che vengono generati usando, quasi sempre, solo le funzionalità di LaTeX[1]

Le funzionalità più interessanti sono

1. la possibilità di rendere i file interattivi mediante l'inserimento di codice JAVASCRIPT (ADOBE SOLUTIONS NETWORK, 2001);
2. la possibilità di comunicare dati ad un server utilizzando un file di dimensioni contenute (FDF)[2] (ADOBE SOLUTIONS NETWORK, 1999).

I pacchetti principali di acrotex sono

1. exerquiz per la generazione di esercizi interattivi (Story, 2012);

*Il lavoro si inserisce nel progetto di ricerca di interesse d'Ateneo dell'Università Cattolica "Progetto M.In.E.R.Va.: Creazione di esercizi di tipo interattivo con percorsi differenziati per il recupero delle carenze e la valorizzazione delle eccellenze in matematica".

1. Per alcuni aspetti, che però non ci interessano, è invece necessario processare il file con Adobe Professional.
2. Attualmente Adobe spinge per l'utilizzo di altri tipi di file, ottenibili con un software a pagamento, ma poiché la possibilità di creare file FDF fa parte del protocollo PDF, dovrà sempre essere implementata.

2. web che gestisce gli aspetti "estetici" (pannelli, titoli, ecc.);
3. eqexam per la generazione di esami;
4. insdljs e dljslib per l'inserimento del codice JAVASCRIPT in documenti LaTeX;
5. eforms per la gestione dei *forms* PDF direttamente in LaTeX;
6. eq2db per la comunicazione PDF-server e la memorizzazione degli esiti sul server;
7. altri pacchetti minori.

Per i nostri scopi, il pacchetto più interessante è exerquiz, che definisce tre ambienti per gli esercizi:

1. problem o exercise che servono per esercizi da svolgere "su carta" perché prevedono una risoluzione estesa;
2. shortquiz che permette di scrivere esercizi a risposta aperta o chiusa, che non prevedono però la comunicazione con il server e l'attribuzione di un punteggio;
3. quiz che funziona come shortquiz, con la differenza che gli esercizi sono valutati e i dati possono essere registrati su un server.

Gli ambienti shortquiz e quiz possono contenere le stesse tipologie di esercizi:

- domande a scelta multipla (MCQ) con una sola risposta esatta;
- domande a risposta multipla (MAQ) in cui ci possono essere più risposte esatte;
- domande a risposta aperta, in cui la risposta, che deve essere scritta rispettando alcune regole, viene esaminata e valutata automaticamente utilizzando codice JAVASCRIPT.

Tutte le domande possono avere punteggi diversi e nelle domande a risposta chiusa ogni risposta può avere un punteggio diverso, anche negativo. Il codice

```
\item
 Nella frase ``Paolo legge un libro'',
 \emph{un libro} è
\begin{answers}{numero colonne}
    \bChoices[random]
\Ans[1]{1} complemento oggetto \eAns
\Ans[0]{0} complemento \eAns
\Ans[-1]{0} soggetto \eAns
      \eChoices
\end{answers}
```

produce in output

Nella frase "Paolo legge un libro", *un libro* è

1. complemento
2. complemento oggetto
3. soggetto

I numeri indicati tra parentesi quadra sono opzionali ed indicano il punteggio attribuito ad ogni risposta.

Il pacchetto prevede di fornire all'utente diversi feedback:

- nell'ambiente shortquiz viene segnalato se la risposta è giusta o sbagliata e si può inserire la soluzione. Inoltre, per le domande a risposta aperta, è possibile, mediante il comando `\tallybox`, inserire una casella accanto a quella della risposta, nella quale vengono contate le risposte sbagliate. Tutti i messaggi di feedback vengono dati immediatamente, appena si inserisce la risposta.
- Nell'ambiente quiz si ha la segnalazione della risposta giusta o sbagliata, la soluzione e il punteggio. Tutte i feedback vengono dati alla fine dell'esercitazione.

Utilizzando eq2db e l'ambiente quiz, alcuni dati possono essere inviati ad un server: i dati anagrafici, le risposte date, il punteggio, la data e l'ora di inizio e fine della prova.

acrotex è stato usato in vari progetti per la somministrazione di esercizi online, come Acroweb (Maříк), MacQTeX (Griffin), ecc..

3 Il nostro lavoro

Il progetto è rivolto a studenti delle scuole medie superiori e lo scopo principale è quello di mettere a disposizione dei percorsi di recupero delle carenze o di consolidamento delle conoscenze che lo studente può fruire in autonomia o sotto la guida del docente.

Nel citato meeting abbiamo illustrato le modalità di realizzazione della parte teorica e delle esercitazioni guidate relative a questo percorso (Messineo e Vassallo, 2012b).

Quest'anno, il nostro lavoro si è concentrato sulla modifica e integrazione di quanto già previsto in acrotex per la realizzazione delle esercitazioni.

3.1 Gli output

Il pacchetto prevede la realizzazione di diverse tipologie di output: l'esercitazione non tracciata, l'esercitazione tracciata e il compito.

3.1.1 L'esercitazione non tracciata

Si tratta di un insieme di esercizi che vengono assegnati allo studente da svolgere autonomamente, a casa o a scuola, e che non prevedono il tracciamento delle risposte come le altre due modalità.

Funzionano come le esercitazioni preparatorie agli esami ECDL. Possono essere assegnate agli

studenti in vario modo (invio per mail, su cd, ecc.). L'idea è quella di rendere disponibili agli studenti set di esercizi, con correzione automatica, che contengano le soluzioni, i richiami alla teoria e dei suggerimenti per arrivare alla risposta esatta.

Questa tipologia di output viene creata caricando il pacchetto minerva con le opzioni esercitazione e offline.

L'ambiente ideale da usare in questo tipo di esercitazione è shortquiz, presentato nel paragrafo 2, poiché è possibile somministrare domande sia a risposta aperta che chiusa, la correzione delle risposte avviene immediatamente, è possibile inserire la soluzione completa[3], si può usare, per le domande a risposta aperta, il comando `\tallybox` ed è possibile fornire dei suggerimenti (per una descrizione dettagliata, si vedano i paragrafi 2 e 3.2).

3.1.2 L'esercitazione tracciata

In questo caso, il docente somministra agli studenti un'esercitazione, ma vuole registrare sul server le risposte date dallo studente e il punteggio ottenuto.

Viene creata caricando il pacchetto minerva con le opzioni esercitazione e online, che illustreremo nel paragrafo 3.2.

In questo caso, l'ambiente migliore da usare è quiz, che, a differenza di shortquiz, prevede sia la possibilità di tracciare le risposte sia quella di avere la correzione del lavoro fatto solo alla fine dell'esercitazione e non appena data la risposta come nell'altro caso.

Anche in questo tipo di esercitazione si possono usare domande a risposta aperta o chiusa, dare suggerimenti e anche la soluzione completa.

Questa tipologia di esercitazione potrebbe essere usata anche in autonomia dagli studenti, per svolgere esercizi di recupero delle carenze o di consolidamento delle conoscenze. A tale scopo, è stato creato un eserciziario online che illustreremo nel paragrafo 3.4.

3.1.3 Il compito

In questo caso, il docente vuole preparare una prova di valutazione da somministrare ad uno o più studenti dello stesso gruppo.

Questo tipo di output viene creato caricando il pacchetto minerva con l'opzione compito, che prevede in automatico l'opzione online.

Questa opzione disabilita la correzione automatica delle risposte, la soluzione e gli eventuali suggerimenti (trattandosi di un compito, nessuno di questi elementi è utile).

Tutti i dati di questa tipologia di prova vengono registrati su un server.

3.2 I nuovi comandi

Il nostro lavoro si è concentrato su estensioni delle funzionalità di exerquiz e di eq2db e della gestione dei dati sul server. Ci siamo basati su quanto già fatto nei pacchetti esami, distribuito su CTAN (Messineo e Vassallo, 2012a), esami-online e minerva, privati (Messineo e Vassallo, 2012b).

Il pacchetto è stato pensato per esercizi di Matematica, ma può essere utilizzato facilmente anche per altre discipline.

Il nostro obiettivo era quello di estendere le caratteristiche presentate, introducendo

1. integrazione nel pacchetto di tutte le funzionalità del pacchetto acrotex;
2. inserimento di parametri e di altri elementi aleatori negli esercizi;
3. possibilità di selezione degli esercizi da un database;
4. possibilità di creazione di prove "personalizzate" (prove diverse per ciascuno studente o prove ripetute sempre diverse per lo stesso studente) in modo automatizzato usando gli elementi aleatori[4];
5. generazione di una copia cartacea delle prove (sia per backup, sia per poter somministrare la prova agli studenti in caso di indisponibilità tecnica della piattaforma);
6. automazione di alcuni meccanismi a seconda della modalità scelta (esercitazione tracciata online, esercitazione offline, compito o esame);
7. nuove funzionalità (suggerimenti);
8. comunicazione al server di un numero maggiore di dati;
9. tracciamento completo degli esercizi assegnati e degli esiti per finalità statistiche e di controllo.

Gli obiettivi citati sono stati perlopiù raggiunti utilizzando solo codice LaTeX. Non così gli obiettivi riguardanti la comunicazione con il server. Gli aspetti relativi al server sono ancora in lavorazione.

In particolare

1. sono state integrate nel pacchetto minerva alcune funzionalità di acrotex non previste nella prima versione. In particolare, abbiamo previsto

3. L'inserimento della soluzione era già previsto nel pacchetto exerquiz, ma nel nostro ci sono stati dei problemi perché il testo dell'esercizio e della sua soluzione è contenuto nel comando `\newproblem`, che, a nostra conoscenza, non poteva contenere `verbatim` come argomento. Abbiamo risolto questo problema mediante l'uso del pacchetto cprotect.

4. È ovviamente possibile anche creare una prova unica da somministrare a tutti gli studenti di una classe o di un gruppo

- la possibilità di inserire, gestire e personalizzare il pannello laterale con i comandi di navigazione;
- la possibilità di inserire domande a risposta aperta o a risposta chiusa con più di una risposta esatta[5];
- la possibilità di assegnare ad ogni risposta (per le domande a risposta chiusa) un punteggio diverso, anche negativo;
- il comando `\tallybox`, per mostrare allo studente il numero di risposte errate dato alla domanda.

Queste nuove introduzioni hanno richiesto, oltre ad una parziale modifica del codice per renderlo compatibile con quanto già inserito fino a quel momento, un attento lavoro di controllo della compatibilità e di gestione di eventuali problemi. Infatti, il nostro meccanismo di rotazione e il fatto che si voglia produrre sempre ed in automatico un file cartaceo hanno spesso generato dei problemi (di impaginazione, ecc.) che di volta in volta abbiamo dovuto risolvere.

2. le modalità di inserimento di parametri ed altri elementi aleatori è già stata ampiamente descritta in (Messineo e Vassallo, 2012a) e in (Messineo e Vassallo, 2013a);

3. la scelta di esercizi da un database costruito autonomamente è una delle caratteristiche del pacchetto probsoln di Nicola Talbot (Talbot, 2011). Il nostro codice si è ispirato molto a quello, in larga parte modificandolo, perché ci erano necessarie funzionalità diverse.

   Abbiamo già illustrato queste modifiche in (Messineo e Vassallo, 2012a) e in (Messineo e Vassallo, 2012b), riprendiamo qui alcuni punti.

   Gli esercizi possono essere contenuti anche in più file e ogni esercizio è l'unico argomento del comando `\newproblem` che compone gli esercizi. La sintassi per la scrittura degli esercizi è identica a quella usata in acrotex, mentre per l'utilizzo dei parametri aleatori e le operazioni con i parametri si rimanda a (Messineo e Vassallo, 2012a). Nell'idea originaria, ogni file è composto da più varianti di esercizi sullo stesso argomento, ma ora ciò non è più necessario. All'interno del file che viene compilato ci sono alcuni comandi che permettono di estrarre dai file degli esercizi le varianti desiderate. I principali comandi, alcuni dei quali sono stati presentati in (Messineo e Vassallo, 2012a) e in (Messineo e Vassallo, 2012b), sono

   (a) `\esercizi{file1,file2...}` che estrae una variante da ogni file, mescola gli esercizi scelti e li manda in output;

   (b) `\estraies[m]{file1,file2,...}` che sceglie aleatoriamente $n - m$ file tra gli n che ha come argomento, estrae una variante da ognuno di essi e le manda in output;

   (c) MNdb, un ambiente introdotto per estrarre m esercizi degli n presenti in un file. Ha come argomento il nome di un "database" di esercizi. In questo database gli esercizi vengono inseriti con il comando `\selectrandomlyn{m}{file}`, che sceglie casualmente m varianti di esercizi da un file. Tale comando può essere usato più volte (applicato a file diversi) all'interno dell'ambiente. La chiusura dell'ambiente manda in output gli esercizi.

   I tre comandi di scelta e l'ambiente MNdb si basano su una versione modificata dei dataset e dei comandi di scelta del pacchetto probsoln. Il meccanismo di scelta e di rimescolamento degli esercizi è stato implementato da noi.

   Questi comandi possono essere usati più volte all'interno di una prova e devono essere inseriti all'interno degli ambienti shortquiz o quiz.

4. La possibilità di avere prove "personalizzate" è stata implementata con un meccanismo già usato in esami-online (Messineo e Vassallo, 2012b). Gli elementi aleatori dipendono da due quantità: la data della prova e un identificativo dello studente (la matricola o un codice assegnato).

   Per la creazione delle prove compiliamo un file che usa il pacchetto datatool di Nicola Talbot (Talbot, 2013) per la lettura e la manipolazione dei file `csv`. Viene letto l'elenco dei partecipanti alla prova, per ogni studente viene creato un file con i dati, viene rinominato il file della prova e viene lanciata la compilazione, che avviene in più passi: si compila prima la versione cartacea, poi quella online e infine il file delle soluzioni. È possibile che questo procedimento possa essere reso più veloce utilizzando un file di formato precompilato, ma tale possibilità deve essere ancora esplorata. Se lo stesso file con gli stessi esercizi viene compilato per lo stesso studente in date diverse, si ottengono prove diverse.

   Se la prova che si vuole compilare è invece unica per tutti gli studenti o per un gruppo, il file che viene compilato non usa datatool e crea un file unico, con l'anagrafica non compilata (come invece avviene per le prove individuali).

5. L'inserimento di questo tipo di domande comporta alcuni problemi di formattazione dei dati registrati sul server, dei quali parleremo nel paragrafo 5

5. La copia cartacea della prova si è resa necessaria per poter somministrare il compito allo studente anche in caso di malfunzionamento del server. Questa esigenza ha creato alcuni problemi: infatti, anche se acrotex prevede la possibilità di creare un output cartaceo, il nostro meccanismo di rotazione degli esercizi, di scelta da un file e la volontà di avere una stringa delle soluzioni ci ha portato a modificare alcuni comandi di acrotex, per cui le funzionalità del pacchetto hanno dovuto essere tutte verificate nuovamente.

   Ad esempio, in acrotex c'è l'opzione proofing che inserisce nelle domande a risposta chiusa un simbolo accanto alla risposta esatta. Noi abbiamo legato a tale simbolo un meccanismo di `\label` e `\ref` che riesce a tenere traccia degli esercizi scelti nei file, della risposta corretta e a trascrivere questi dati nel file delle soluzioni e nel file aux. Questo comporta che l'opzione sia sempre attiva e quindi, ad esempio, che nell'ambiente shortquiz ci sia sempre il feedback immediato.

6. l'automazione dei meccanismi a seconda della modalità scelta è stata ottenuta introducendo alcune opzioni per la creazione automatica di diversi elementi. Con l'opzione esercitazione viene prevista la possibilità di avere la correzione degli esercizi, di vedere le soluzioni ed eventualmente di avere dei suggerimenti. Sono previste inoltre due ulteriori opzioni, online e offline, che permettono di salvare o non salvare i dati sul server. Inoltre, in modalità offline è possibile usare l'ambiente shortquiz nel quale lo studente ha un feedback immediato alle risposte date.

   Con l'opzione compito vengono disabilitati tutti i suggerimenti e le correzioni e i dati vengono sempre trasmessi al server.

7. La possibilità di avere dei suggerimenti è l'unica variazione di acrotex visibile allo studente (oltre a quella di avere i campi anagrafici precompilati se l'esercitazione è "personalizzata"). La generazione dei suggerimenti è ottenuta mediante il pacchetto fancytooltips di R. Mařík (Mařík, 2012) che inserisce nel file PDF delle pagine di un file PDF esterno come icone nascoste rese visibili mediante il passaggio (o il click) del mouse. I suggerimenti vengono generati in un file esterno.

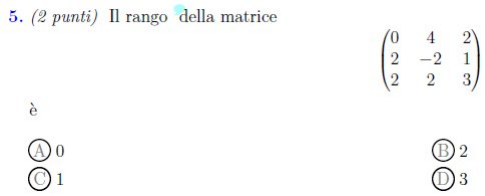
   Il codice per creare un file esterno di suggerimenti è

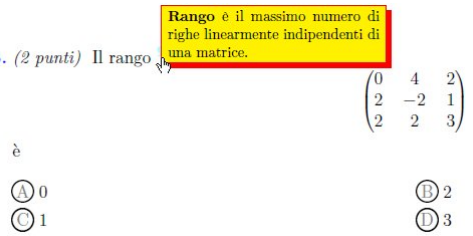
```
\documentclass{article}
\pagestyle{empty}
\usepackage[createtips]{fancytooltips}
\usepackage{fancybox}
\parindent 0 pt
\usepackage{xcolor}
\usepackage{multido,graphicx}
\usepackage[papersize={22cm,16cm},
margin=1pt]{geometry}
\long\def\suggerimento#1#2{
\setbox0=\hbox{\begin{minipage}{2in}
\fboxsep 0 pt
\color{red}
\shadowbox{
{\fboxsep 4pt\colorbox{yellow}
    {\begin{minipage}{\linewidth}
      \color{black}#2
    \end{minipage}}}}
\end{minipage}\ \ \ \ }
\pdfpagewidth=\wd0
\pdfpageheight=\ht0
\advance \pdfpageheight by \dp0
\copy0
\keytip{#1}
  \newpage
}
\begin{document}
\suggerimento{rango}{\textbf{Rango} è
il massimo numero di righe linearmente
indipendenti di una matrice.}
\end{document}
```

Il codice per richiamare i suggerimenti nel file principale è

```
\item\PTs{2} Il rango \tooltip*{ }{1}
della matrice
\[ \begin{pmatrix}0 & 4 & 2\\
2 &-2&1\\
2 & 2& 3\\
\end{pmatrix}
\]
è
\begin{answers}{2}
\bChoices
\Ans0 0 \eAns   \Ans1 2 \eAns
\Ans0 1 \eAns   \Ans0 3 \eAns
\eChoices
\end{answers}
```

Questo codice produce il seguente output





8. Per quanto riguarda le comunicazioni con il server, ci si è mossi in due direzioni: utilizzare (lato server) altri linguaggi oltre ad ASP, già implementato in acrotex, e ampliare la quantità di informazioni trasmesse.

   Il pacchetto acrotex prevede l'utilizzo di un server ASP e di librerie specifiche che la Adobe ha creato per Windows per la lettura dei file FDF. In realtà, il file FDF è un file di testo (con un'intestazione binaria) e può quindi essere letto in più modi (la Adobe stessa mette a disposizione le librerie PERL e JAVA). Per questo motivo, si sono realizzati degli script in linguaggio ASPX e PHP (si veda il paragrafo 3.4). Questo ha portato a dover modificare alcuni file del pacchetto eq2db in modo da ottenere che le comunicazioni con il server avvenissero in modo corretto.

9. Per quanto riguarda le informazioni che vengono inviate al server, abbiamo implementato con l'aiuto di D.P. Story due procedure in linguaggio ASP: la prima invia, oltre a tutte le altre informazioni, anche la risposta esatta alle domande proposte; la seconda memorizza, salvandolo in un file, il contenuto del file FDF[6]. Queste modifiche rispondono a due necessità: da un lato, poter salvare il file FDF permette, attraverso l'unione di tale file con il corrispondente PDF, di ricreare la prova come è stata svolta dallo studente[7] e ciò risponde ad ogni tipo di esigenza di conservazione delle prove di valutazione; dall'altro, questo è il primo passo verso l'obiettivo di raccolta e analisi dei dati per poter valutare l'efficacia degli esercizi proposti sia per il miglioramento degli studenti che per discriminare il loro livello di preparazione.

3.3 Una breve digressione sul riordinamento di liste e scelta casuale

Nel pacchetto minerva gli elementi casuali intervengono in molti punti (scelta degli esercizi, riordinamento delle domande, delle risposte, ecc.). Questi elementi casuali si basano su tre diversi metodi di riordinamento di una lista (e molti altri metodi sono presenti in altri pacchetti). Abbiamo deciso di mantenere tutti e tre i metodi per diversi motivi:

- una maggiore semplicità di sviluppo dei comandi per la scelta sulle diverse liste di oggetti in quanto questi fanno riferimento a quei metodi di scelta
- la possibilità di avere una maggiore variabilità nella generazione delle prove date le diverse procedure di scelta (se il metodo per la scelta degli esercizi porta ad avere lo stesso esercizio in due prove diverse, è estremamente probabile che l'ordine delle risposte sia diverso proprio per la differente procedura di scelta)
- la differenza di gestione di uno dei metodi rispetto agli altri due: infatti il metodo che viene usato (anche in acrotex) per la rotazione delle risposte nelle domande a risposta chiusa si basa su dei tag e permette di tenere degli elementi fissati.

Vorremmo qui analizzare brevemente questi tre metodi.

Il primo metodo è una nostra variazione di un codice trovato in rete. Il comando è definito usando il pacchetto xargs che permette di avere più parametri opzionali in un comando e di averli in qualunque posizione (nel nostro caso il parametro opzionale è il terzo).

```
\newcommandx{\rand@getitems}[3][3=item]
{% #1=numero totale elementi lista #2=
% nome file arrivo #3=nome file partenza
  \i@sh=#1% %\theshuf@tot
  \loop%
\expandafter\let
\csname flag\number\i@sh\endcsname a%
    \advance\i@sh by-1%
    \ifnum\i@sh > 0 \repeat%
  \i@sh=#1% %\theshuf@tot
  \loop%
    \setrannum{\j@sh}{1}{#1}%
\expandafter\ifx
\csname flag\number\j@sh\endcsname a%
\expandafter\let
\csname flag\number\j@sh\endcsname b%
 %Controllo se è già stato scelto
\FPeval\n@tmp{round(#1-\number\i@sh+1:0)}
 \expandafter\edef
 \csname #2\romannumeral\n@tmp\endcsname
 {\csname #3\romannumeral\j@sh\endcsname}
\advance\i@sh by-1%
    \fi%
    \ifnum\i@sh>0 \repeat
}
```

Il comando funziona in questo modo: con una procedura precedente è stato assegnato a ogni elemento di una lista un comando che è dato da un nome + un numero romano, di default \itemi, \itemii,

6. Un'analoga procedura è stata implementata anche in ASPX e in PHP.

7. Per ora tale unione è possibile solo manualmente usando Adobe Reader e non in via automatica con programmi come Pdftk o simili.

. . . ; la stessa procedura conta gli elementi della lista `\theshuf@tot`. Ora si sceglie casualmente un numero dell'elenco (e quindi l'elemento della lista che ha quel numero) e si assegna a una nuova variabile (ad esempio `\casoi`) l'elemento scelto e si prosegue così, controllando ad ogni passo se il numero scelto casualmente era già stato scelto (controllo sui flag).

Il secondo metodo è una nostra variazione del codice per mescolare gli elementi di una lista usato in probsoln

```
\newcommand{\shuffle}[3][]{%
\@shfctr=1\relax
\whiledo{\@shfctr < 101}%
{%
\setrannum{\@shfA}{1}{#3}
\setrannum{\@shfB}{1}{#3}
    \ifnum\@shfA=\@shfB
    \else
\edef\@@tmpA
{\csname#2\romannumeral\@shfA\endcsname}
      \let\@tmpA=\@@tmpA
\edef\@@tmpB
{\csname#2\romannumeral\@shfB\endcsname}
      \let\@tmpB=\@@tmpB
\expandafter\xdef
\csname#2\romannumeral\@shfA\endcsname
{\@tmpB}
\expandafter\xdef
\csname#2\romannumeral\@shfB\endcsname
{\@tmpA}
    \fi
    \advance\@shfctr by 1\relax
    \ifthenelse{\equal{#1}{}}{}{
  \@shfA=0%
  \loop
  \advance\@shfA by1
\expandafter\xdef
\csname#1\romannumeral\@shfA\endcsname
{\csname#2\romannumeral\@shfA\endcsname}
  \ifnum\@shfA<#3 \repeat
}
  }%
}
```

Questo comando funziona in pratica con lo stesso meccanismo del mescolamento delle carte: dati gli elementi di una lista a cui come prima è stato assegnato un "nome" e stabilito il numero di elementi della lista, si scelgono due numeri casuali nell'intervallo assegnato e, se diversi, si scambiano tra loro; la procedura viene eseguita per un certo numero di volte (nel nostro caso 100) e alla fine si ha una lista riordinata a cui vengono riassegnati gli elementi originali.

Il terzo metodo è quello usato nel pacchetto acrotex per la rotazione delle risposte nelle domande a risposta chiusa.

```
\def\aeb@randomizeChoices#1{%
    \setrannum{\aeb@ranChoice}{1}{#1}
\count0=0
\aeb@hold=\expandafter{\@temphold}
    \def\@temphold{}%
\expandafter\@tfor\expandafter
 \@temp\expandafter:
  \expandafter=\the\aeb@hold \do {%
        \advance\count0by1
  \ifnum\count0=\aeb@ranChoice
  \aeb@hold=\expandafter\expandafter
\expandafter
{\expandafter\@tempholdrandom\@temp}%
\edef\@tempholdrandom{\the\aeb@hold}%
        \else
\aeb@hold=\expandafter\expandafter
\expandafter
{\expandafter\@temphold
\expandafter{\@temp}}
\edef\@temphold{\the\aeb@hold}%
        \fi
    }%
    \aeb@numChoices=#1
 \advance\aeb@numChoices-1
\ifnum\aeb@numChoices=0\relax
\def\@next{\aeb@finishedRandomizing}
\else
\def\@next
{\aeb@randomizeChoices
{\the\aeb@numChoices}}
\fi    \@next
}
\def\aeb@finishedRandomizing{%
\aeb@hold=\expandafter\expandafter
\expandafter
{\expandafter\@tempholdrandom
\@tempholdfreeze}
\gdef\@temphold{}\gdef\@tempholdrandom{}
\gdef\@tempholdfreeze{}%
    \edef\finished@Randomizing{%
    \noexpand\@@ansChoices
    \the\aeb@hold
    \noexpand\eChoices}%
    \finished@Randomizing
}
```

Questa procedura è in un certo senso simile alla prima: si sceglie un numero casuale (e quindi il corrispondente elemento) e lo si mette in una lista "di prescelti"; dalla lista contenente i soli elementi restanti si estrae un nuovo numero casuale e quindi l'elemento corrispondente che viene spostato nella lista dei prescelti e così via. Il procedimento è complicato dal fatto che è possibile definire uno o più elementi della lista come inamovibili e quindi tali elementi devono prima essere eliminati e immessi in un'altra lista e, successivamente, reimmessi nella lista definitiva.

Sarebbe interessante verificare, magari considerando altre procedure, quali siano più efficienti sia in termini di effettiva redistribuzione casuale, sia

in termini di velocità di esecuzione.

3.4 I programmi di supporto

Come illustrato lo scorso anno, l'utilizzo dei materiali prodotti ha reso necessaria la creazione di una piattaforma web attualmente ospitata sul sito dell'IIS "Falcone-Righi" di Corsico (una delle scuole aderenti al progetto) e visibile all'indirizzo `http://minerva.falco.mi.it`. Il server web utilizzato è Microsoft IIS per poter utilizzare gli script in linguaggio ASP forniti con acrotex.

Come già accennato, una parte del progetto ha previsto la realizzazione di script in linguaggio diverso da ASP, sia per rendere la piattaforma indipendente dalle librerie della Adobe, sia per poter migrare la piattaforma su un server APACHE, in modo che l'intero progetto si serva di software Open Source.

Le due modifiche introdotte sono state

1. Un collaboratore della Direzione Sistemi Informativi dell'Università Cattolica ha implementato un parser nel linguaggio ASPX di Microsoft che funziona anch'esso su un server IIS. Tale script funziona senza l'utilizzo di librerie specifiche e salva i dati sia in un file `csv` che in un database. Inoltre, ha la possibilità di salvare il file FDF completo.

2. Uno studente dell'ITI "Lagrange" ha realizzato un parser in PHP, di cui si è parlato in (Messineo e Vassallo, 2013b), per interpretare i dati inviati mediante il file FDF. Questo script salva i dati in un file `csv`, in modo da poterli importare in un database consultabile, attraverso una pagina HTML, dai docenti o dall'amministratore della piattaforma. Lo script riconosce in automatico il tipo di file FDF e la tipologia di dati inviati e li registra di conseguenza. Anche questo script è in grado di salvare il file FDF completo.

I due parser elaborati non dipendono (come invece quello originale di acrotex) da librerie esterne e sono al momento "statici", ovvero è necessario definire all'interno dello script quali campi del file FDF devono essere salvati.

Lo studente dell'ITI "Lagrange" ha creato anche un *eserciziario online*: si tratta di una piattaforma (scritta anch'essa in linguaggio PHP) che consente ai docenti di caricare gli esercizi da loro prodotti, classificandoli secondo alcuni criteri prestabiliti, e di scegliere tra gli esercizi già caricati quelli necessari per la creazione della prova che desiderano somministrare.

In questo eserciziario, un docente può caricare esercizi nuovi. Tali esercizi devono essere scritti in LaTeX, seguendo le istruzioni già descritte in (Messineo e Vassallo, 2012a). Occorre caricare anche uno o due file PDF generati con il file di controllo, `totale-versioni`. Tali file contengono il testo di tutte le varianti degli esercizi in formato numerico e, se possibile, anche in formato parametrico. Si deve anche indicare il numero di varianti, il grado di difficoltà, la classe e la scuola a cui è rivolto.

È possibile scaricare gli esercizi e generare tutti i file necessari per la preparazione di una prova. Una funzione di preview permette di vedere gli esercizi scelti e di selezionarli; l'inserimento di alcuni dati permette la creazione dei file necessari per la compilazione. La piattaforma permette lo scaricamento di un file compresso, protetto da password, il contenuto dovrebbe permettere a chiunque abbia a disposizione i pacchetti necessari di compilare la prova, senza la necessità di intervenire sui file `tex`. Questo è il primo passo verso la possibilità di una compilazione online delle prove per tutti gli studenti da parte del docente e delle prove di recupero o esercitazione individuale da parte del singolo studente.

C'è infine da notare che lo studente ha fatto in modo che l'eserciziario sia visibile e, almeno in parte, utilizzabile con qualsiasi dispositivo, dallo smartphone al desktop.

4 La sperimentazione

Come accennato, il pacchetto è stato creato nell'ambito di un progetto di ricerca volto a sperimentare una nuova modalità di recupero delle carenze in Matematica e può essere usato anche per altre discipline.

Il pacchetto, le tipologie di esercizi e la piattaforma sono stati sperimentati in alcune classi dell'IIS "Falcone-Righi" di Corsico, sia mediante la somministrazione di esercitazioni tracciate ed altro materiale di supporto, sia mediante il coinvolgimento diretto degli studenti nella creazione degli esercizi.

La prima parte della sperimentazione è stata presentata lo scorso anno al convegno GUIT (si veda (Messineo e Vassallo, 2012b)). Alcuni risultati preliminari sono stati presentati al convegno Didamatica 2013 (Messineo e Vassallo, 2013b) e in due articoli successivi, (Messineo e Vassallo, 2013c) e (Messineo e Vassallo, 2013d).

5 Questioni aperte

Rimangono ancora alcuni problemi aperti e funzionalità da implementare.

Una funzionalità che vorremmo implementare sulla piattaforma web è la possibilità di modificare online i file con un editor LaTeX e, successivamente, di poter compilare su richiesta direttamente i file PDF sul server.

I problemi aperti riguardano soprattutto la gestione dei file FDF che vengono inviati al server.

Il primo problema è la gestione da parte del codice JAVASCRIPT degli elenchi di risposte nelle domande a risposta multipla: il file PDF manda

al server la lista delle risposte date come elenco separato da virgole; se tra le domande ve ne sono a risposta multipla, la stringa delle risposte a tale domanda è un altro elenco separato da virgole: la gestione di questo elenco all'interno dell'altro elenco non crea problemi per l'attribuzione dei punteggi e l'esito finale (gestito localmente dal codice JAVASCRIPT), ma determina un output scarsamente leggibile. Ad esempio l'output *a,a,b,c,b* riferito a 3 domande a risposta singola e una a risposta multipla può essere dato da *a,{a,b},c,b* o da *a,a,{b,c},b* e per distinguere i due casi è necessario confrontare la stringa con l'elenco delle domande (la situazione si complica se non vengono date delle risposte).

Il secondo problema è relativo all'archiviazione dei dati: come si è detto la coppia "file PDF" (testo dell'esame) e "file FDF" (risposte date) può essere unita in un unico file PDF che contiene così la prova completa dello studente. Questa procedura può essere automatizzata anche su più file e può essere eseguita sul server tramite il software PDFTK: purtroppo al momento questo non è possibile per qualche problema di compatibilità del file FDF, e la procedura può essere eseguita solo utilizzando Adobe Reader su ogni singolo file: si tratta quindi di vedere quale sia il problema che impedisce l'uso della procedura automatizzata.

Il terzo problema riguarda il tracciamento dei dati a fini statistici: la nostra idea è quella di poter utilizzare il database in cui vengono salvati gli esiti a fini statistici per analizzare sia i progressi degli studenti su ogni singolo argomento, sia per verificare se le domande dei test siano o meno efficaci a discriminare il livello degli studenti. Attualmente noi salviamo solo gli esiti di ogni singola prova per ogni studente, quindi possiamo analizzare i progressi globali dello studente (o della classe). Alcuni di questi dati sono salvati nel file `aux` proprio per il meccanismo di `\label` e `\ref` adottato e da lì riportati nel file `tex` delle soluzioni

```
\item
\ref{a-e:20130507-file:equadue-NPE-q:xix}
\item
\ref{a-e:20130507-file:equadue-NPE-q:vii}
item
\ref{a-e:20130507-file:prova-aperto-q:i}
```

dove *a* sta per *answer*, *e:20130507* indica l'ID dello studente, *file:equadue-NPE* indica il file degli esercizi da cui si estrae la domanda, *q:xix* indica il numero della domanda nel file. Si tratta quindi di riuscire a riportare questi dati sul server (se possibile con il file FDF) e immetterli nel database in modo da poter sapere a quale domanda lo studente XY ha risposto esattamente, quanti studenti hanno risposto a una certa domanda ecc. Lo scopo finale è quello di riuscire a realizzare un'analisi dettagliata dell'efficacia della domanda nel testare le abilità degli studenti, usando una metodologia, analoga a quella che viene utilizzata per l'analisi delle risposte delle prove INVALSI, basata sul modello di Rasch (Bond e Fox, 2007).

6 Conclusioni

Il progetto M.In.E.R.Va., di cui il pacchetto minerva fa parte, pur essendo ancora in fase sperimentale, si è dimostrato un ausilio molto efficace per la didattica della Matematica nelle classi in cui è stato usato.

L'obiettivo finale è quello di realizzare una piattaforma che possa essere usata con facilità sia dai docenti con nessuna conoscenza di LaTeX (che quindi fruiscono dei percorsi già predisposti o scelgono gli esercizi dal database e devono solo compilare e caricare la prova), sia da docenti "esperti" (che possono anche inserire nuovo materiale o personalizzare quello esistente), sia da studenti, che possono scegliere in autonomia i percorsi o gli esercizi da eseguire. Inoltre, lo strumento deve mettere a disposizione di tutti gli utenti alcune statistiche significative.

Il nostro auspicio è quello di avere a disposizione una piattaforma sufficientemente robusta da consentire anche la compilazione "al volo" delle prove, sul modello, ad esempio, di MacQTeX (Griffin).

I file necessari per l'utilizzo del pacchetto e le comunicazioni con il server, con una minima documentazione, sono al momento disponibili al link `https://www.dropbox.com/sh/jf4miydsrd4ymub/LqqutgJk5S`, in attesa di una versione definitiva che sarà rilasciata su CTAN.

I file necessari per la comunicazione con il server sono disponibili in ASP (estratti dal pacchetto acrotex con alcune nostre modifiche), in ASPX (opera di Bruno Cibelli, che ha collaborato con noi alla realizzazione di parte del progetto), in PHP (opera di Mohamed Abedalla, studente dell'ITI "Lagrange" di Milano), non completamente testato.

La piattaforma usata presso l'IIS "Falcone-Righi" è visibile all'indirizzo `http://minerva.falco.mi.it`. È possibile visualizzare un percorso demo o chiedere la registrazione come docenti di prova per sperimentarne l'uso.

Riferimenti bibliografici

ADOBE SOLUTIONS NETWORK (1999). *FDF Toolkit Overview Technical Note # 5194*. ADOBE SYSTEMS INC.

— (2001). *Acrobat Forms JavaScript Object Specification, Version 5.0.5 Technical Note # 5186*. ADOBE SYSTEMS INC.

Bond, T. G. e Fox, C. M. (2007). *Applying the Rasch Model: Fundamental Measurement in the Human Sciences*. Psychology Pr.

Griffin, F. «MacQTeX». URL `http://rutherglen.ics.mq.edu.au/~macqtex/`.

Maříк, R. «Acroweb». URL http://user.mendelu.cz/marik/index.php?item=41.

Maříк, R. (2012). «Il pacchetto fancytooltips». CTAN:/macros/latex/contrib/fancytooltips/.

Messineo, G. e Vassallo, S. (2012a). «Il pacchetto esami per la creazione di prove scritte». ArsTEXnica, **14**, pp. 95–103.

— (2012b). «Test online di matematica: il progetto M.In.E.R.Va». ArsTEXnica, **14**, pp. 104–112.

— (2013a). «The esami package for examinations». *TUGboat*, **34** (1), pp. 40–46.

— (2013b). «Il progetto M.In.E.R.Va». In *Atti del Convegno Didamatica 2013*. pp. 689 – 698.

— (2013c). «Il progetto M.In.E.R.Va». *Nuova secondaria*, pp. 93–95.

— (2013d). «Il progetto M.In.E.R.Va per il recupero e la valutazione». *Bricks*. URL http://bricks.maieutiche.economia.unitn.it/?cat=239.

Story, D. P. (2012). «Exerquiz & AcroTEX». URL http://www.acrotex.net/.

Talbot, N. L. (2011). «Il pacchetto probsoln». CTAN:macros/latex/contrib/probsoln.

— (2013). «Il pacchetto datatool». CTAN:macros/latex/contrib/datatool.

▷ 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

Experiments with multitasking and multithreading in LuaTeX

Luigi Scarso

Abstract

We describe a SWIG wrapper to the Pthreads and ZeroMQ C libraries and how they can be used to add multitasking and multithreading features to the Lua interpreter of LuaTeX. Simple examples are shown.

Sommario

Presentiamo un *wrapper* SWIG a Pthreads e le librerie C ZeroMQ e il modo di usarli per aggiungere le proprietà di *multitasking* e *multithreading* all'interprete Lua di LuaTeX. Mostriamo anche dei semplici esempi.

1 Introduction

The average TeX user currently uses a recent TeX distribution on a quite powerful hardware, usually a multicore, Intel-based desktop or a notebook PC. The operating system is very likely (in alphabetical order) Linux, OS X or Windows. Latest releases of all of them support these Intel CPUs, so a natural question is "Can LuaTeX have some gain if it supports these additional processors?" Before answering the obvious way, we will consider two kinds of support. In the first one, that we call *implicit*, the program source code (in our case LuaTeX written in C) is modified by translating, whenever possible, a sequential chunk of code into an equivalent parallel one: e.g., supposing that `a` is an array of 8 integers, the `for` loop

```
for(int i=0;i<8;i++)
  a[i]=2*i;
```

can be executed in parallel with a max speed up $S = 8\times$ if we have at least eight processors.

The programmer usually has to indicate the use of a parallel `for` to the compiler (for example with `pfor`) and let the compiler translate the loop body into parallel code. We can immediately see pros and cons of the internal support: the end user (the TeX user) doesn't have to know any new instruction (i.e., TeX macros are still valid) to gain the support of extra processors but, on the other side, the same user needs a compiler that supports a minimal set of "parallel constructs" which work at least under Linux/OS X/Windows. This last point is the least difficult: OpenMP (*Open MultiProcessing*) is a C library that offers some parallel constructs by `#pragma` directives, so if a compiler doesn't support these directives it's still able to compile the code. For example the aforementioned loop statement can be translated in OpenMP in

```
#pragma omp parallel for
for (i = 0; i < 8; i++)
  a[i]=2*i;
```

OpenMP is supported by the following compilers: GNU `gcc`, Intel `icc` and Microsoft `cl.exe`, while the `LLVM` support is ongoing. OpenMP is a well-established and supported library, with a good community and a rich documentation (see for example Chapman *et al.* (2007)) but it doesn't automatically reveal the internal parallelism of a program. This task requires a huge amount of resources in terms of man-hour and the benefits can be less than expected. Keep considering the `for` loop example already seen: while it's reasonable to expect that each `2*i` operation is performed in parallel, how are memory assignments `a[i]=2*i` performed? They require a shared memory which is, as a consequence, a shared resource requiring a synchronized access. This access can in turn reduce the parallelism.

Even if we think of an architecture with some amount of private memory, many situations require the *communication* of data between different processors and hence again a synchronized access to a shared resource (the bus of communication in this case). Different hardware implementations can show different performances with the same code.

Another important point is the Amdahl's law to calculate the speed up S:

$$S = \frac{T_{\text{seq}}}{T_{\text{par}}} = \frac{1}{(1 - F_e) + F_e/S_e}$$

where T_{seq} and T_{par} are the execution time for the sequential version and for the parallel version of the program, F_e is the fraction of the original time in the sequential execution that can be converted in a parallel one and S_e the speed up that can be obtained if *all* the sequential code can be converted into a parallel one (Hennessy e Patterson, 2012, p. 46). So, for example, if we have eight processors and want a speed up $S = 4$, it might seems reasonable that only 50% of the code need to be rewritten. Amdahl's law says instead that $4 = \dfrac{1}{(1 - F_e) + F_e/8} \Longrightarrow F_e = 6/7 \approx 86\%$ of the code needs to be rewritten. If we just rewrite

50% of the same code, we will have a mere speed up of $S = 1.77$.

The second way to support more than a processor is to *explicitly* delegate the end user to manage concurrent jobs. It's better to explain now that the difference between parallelism and concurrency consists in timings: "a system is said concurrent if it can support two or more actions in progress at the same time. A system is said to be parallel if it can support two or more actions executing simultaneously" (Breshears, 2009, p. 2). According to the previous definition, a system with one processor can be concurrent but it is not parallel; a system with at least two processors can be parallel (and hence concurrent).

It can be a bit surprising but, since all of the aforementioned OSs support background execution, often by means of the system shell, a simple solution for a concurrent TEX is to run in background the needed jobs: the `shell_escape` environment variable must be enabled and some macros have probably to be written to encapsulate the specific call in the system shell. The problem is the communication of data between different jobs, that can be inefficient or even impossible, and the fact that there is a poor control over the job themselves (for example, it can be impossible to kill a background job).

In this paper we will consider how to use LUATEX to explicitly manage parallel/concurrent jobs, so we will not consider the OpenMP library. In the next sections we will first introduce a minimal terminology, then we will present the used tools, then we will conclude showing some simple examples not specifically tailored for LUATEX.

2 Processes, threads and processors

At the very beginning of the computer's age, a typical computer had an expensive CPU and central memory, both limited in speed and size. This means that the first Operating Systems (*OS*) were not too complex; even the work organization had to be quite elementary: each job had to be executed in a serial way (*batch processing*) almost without user interaction. This will be the optimal organization (i.e., it maximizes CPU and memory usage) if each job has no interaction with Input/Output (*I/O*) devices; if a job had a long session with a slow I/O (like a long printing) the CPU would have to wait until the end of the session while it could be profitably used to serve one of the waiting jobs. With the increasing power of CPUs and memory size, the next step was the creation of a primitive OS to manage more than a job still executing a single job at time: if a job is waiting the end of an I/O operation, the OS saves (a representation of) this job in the memory and gets the next job; when an operation ends, the associated job has to wait until the processor is again assigned to it. This technique is called *multiprogramming*.

Before going on, it is better to remind some terms. A *programming language* is a *formal language* that has a machine as target; a *computer* programming language has a computer as target. An alphanumerical string is called a *program* in a programming language if 1) it is written according to the syntax and the semantic of the given programming language and 2) it correctly translates an algorithm. The machine (i.e. a computer) running the program progressively reveals the underneath semantic *executing* program statements, that we can consider more elementary pieces of the program itself. The execution of a program requires that the OS creates and fills the appropriate data structures in the memory so that the CPU can find the machine instructions to execute. When the execution ends, data structures are erased, freeing the memory for another execution. Data along with CPU registers form the *task* or *process* associated to the program. The term process is generally used in the context of the data structures for a specific OS, while task is used in broader sense. Anyway, both of them indicate a *dynamic* entity, i.e an activity that has a starting and a running time and that, under certain circumstances, can be stopped and restarted.

An OS is said *multitasking* (seldom *multiprocessing*) if it supports more than one task at the same time. There are two kinds of multitasking: *cooperative*, where the task itself returns control to the OS or releases resources, and *preemptive*, where the OS will manage the execution of tasks, stopping one of them after a fixed amount of time (*quantum* or *time slice*), and resumes the run of another task. There is a difference between a multiprogramming and a multitasking OS. The latter is designed to use a single CPU and memory that are fast and large so that interactions with the user is no more negligible. The OS will indeed manage system and users processes to give all an adequate amount of CPU time (*fairness*) and will guarantee an overall good responsiveness. On the other side, like in the multiprogramming, each process still sees the computer as a dedicated CPU with a memory of contiguous addresses starting from zero and ending to some limit (*flat memory model*), eventually higher than the physical memory actually available in the computer, and the OS takes care to keep separate each process space. If a process tries to access an invalid address it will be terminated (*killed*) by the OS. This is a mandatory feature for a multitasking OS, but it comes with a price of big data structures. The time to create a process, or to clone an existing one, is long, as is the switch from a process to another one (*process context switch*). Another complication is the communication between processes (*InterProcess Communications* or IPC) that involves the

OS due to the need to translate memory addresses between processes. A solution to these problems is to allow the existence of "processes" inside the same address space, the *threads*. A process is hence a collection of one or more threads and the key point is that the context switch between threads is faster than between processes (even ten times, see Kerrisk (2010, p. 610) for Linux, but similar figures are also valid for Windows). On the other side, if a thread tries to access a wrong memory address, the entire process and its related threads are killed; the same stands when a thread exits (i.e. signals to the OS that its process is terminated) all the other threads also are killed in case they have finished their run.

At this point we have to consider that a current common desktop computer has more than one processor. More precisely, a *motherboard* can have one, two or four *sockets* hosting a *Central Processing Unit* (CPU), and each CPU can have from two to four *cores*. Each core can manage a different process, and in the most common hardware architecture, the SMP (*Symmetric MultiProcessor*), all cores are peers. A process running in a core can be stopped, moved to another core and resumed. Furthermore, the Intel "Hyper-Threading Technology" (HTT) splits each core into two logic CPUs with shared L1 cache. This is a proprietary implementation of the *Simultaneous MultiThreading* (SMT) technique that maps a thread into a virtual CPU to maximize the use of a core (the rationale is that two threads of the same process share the same address space so the cache L1 and L2 caches have better chances to already have the needed instructions and data). The HTT is a hardware feature and need support from the OS. As examples, computers used for this paper have a K55V single socket motherboard by ASUSTeK COMPUTER INC. hosting an Intel© CORE™i7-3610QM at 2.30GHz with 4 cores which implements the HTT, offering 8 virtual CPUs, and a T101MT single socket motherboard, from ASUSTeK COMPUTER INC. as well, equipped with an Intel© ATOM™ N450 at 1.66GHz with 1 core that still implements the HTT, offering 2 virtual CPUs. The first one is in a notebook running Linux and the second one in a netbook running Microsoft Windows 7 Home Premium[1] and both the CPUs are soldered on the motherboard (so practically there is no socket). Both OSs support SMP and SMT (see Kerrisk (2010) for Linux and Russinovich *et al.* (2012a) and Russinovich *et al.* (2012b) for Windows 7), but under Windows the SMP is not enabled because there is only one core. As we can see, the single processor era is ended.

1. On Linux this information is available using the `dmidecode` command; on Windows, using the `System Information` command

3 Supporting threads and processes under LuaTEX

In the first section we have seen that OpenMP is a widely used library to expose the implicit parallelisms. The question we would like to answer is "Is there something similar for explicit parallelism/concurrence in TEX?" We can consider the following facts:

- TEX has no construct for concurrence but most, if not all, current modern TEX implementations have the `\write18` macro that can execute a system command, so it can be used to launch a program in background;
- Lua (and hence luaTEX) *supports* cooperative multithreading by `coroutines` (Ierusalimschy, 2013, ch. 9) but this doesn't mean that a Lua `thread` can be scheduled by the underlying OS as a user process/thread. Anyway Lua (and luaTEX) can call an external C library when a suitable wrapper is present, and this one *can* create a user process/thread;
- despite Linux, OS X and Windows are different OSs, they share the same concept of process and thread[2] and they are mostly implemented in C, with (usually) Assembly code for drivers and C++ code for some abstract constructs (OS X also uses Objective C). Standard C has no support for processes/threads (only the latest C++11 standard has the new `std::thread` class).

The problem to come is to find a library that at least provides support for threads. Under UNIX©, the most important library is POSIX Threads (Pthreads), which is part of IEEE standard 1003.1 (see `http://pubs.opengroup.org/onlinepubs/9699919799`). Linux has an almost full support for Pthreads and the API are described in Kerrisk (2010, ch. 28–33); OS X fully supports Pthreads (see `http://www.opengroup.org/openbrand/register/brand3591.htm`) and, even if Pthreads are not directly supported in Windows[3], the Pthreads win32 project at `https://sourceware.org/pthreads-win32/` is a ten-years-old project, still active, which offers an almost complete implementation of the API. The other important point is that the API is specified by a set of C header files so it looks reasonable to try to build a Lua wrapper for luaTEX.

3.1 A SWIG wrapper for Pthreads

One of the key points of Lua design is to facilitate interfacing with external libraries with a complete

2. Windows has also a variant of threads, called *fiber*, similar to Lua `coroutines`.

3. Windows has a partial support for POSIX.1 as subsystem, but not for Pthreads.

API. SWIG, the *Simplified Wrapper and Interface Generator*, is a program that assists the developer in building a wrapper and can automatize many aspects. Supposing that header files that list the API are in the `pthread` subfolder of the current folder, a typical workflow consists in preparing an *interface* file `core.i` like this:

```
/*   core.i */
%module core
%{
#include "pthread/sched.h"
#include "pthread/semaphore.h"
#include "pthread/pthread.h"
%}
%include "pthread/sched.h";
%include "pthread/semaphore.h"
%include "pthread/pthread.h"
```

and let `swig` produce the C source of the wrapper, `core_wrap.c` with

```
swig -importall -I/usr/include
 -I/usr/lib/gcc/x86_64-linux-gnu/
    4.7/include/
 -lua core.i
```

Then the source of the wrapper is compiled and linked against the Pthreads library:

```
gcc -fpic -I./pthread -pthread \
     -c core_wrap.c -o core_wrap.o
gcc -Wall -shared -pthread core_wrap.o \
     -llua5.2 -lpthread  -o core.so
```

That's all: the C API are now available to Lua (and luaTEX):

```
local _core= package.loadlib("./core.so",
                          "luaopen_core")
if not(_core) then
  print("error loading _core")
  return 1
end
local pthread = _core()
for k,v in pairs(pthread) do
  print(k,v)
end
```

A complete interface file (for Linux) is given in fig. 2 and 3, but it is better to discuss some aspects of threads, C and Lua before commenting the file.

One of the main problem of processes and threads is the management of a shared resource, which almost always requires a synchronized access. We can think of a shared resource like a cross between fours roads: without synchronized semaphores a crash is almost secure if nobody takes care to respect the rules. With threads, the memory is a shared resource because, as we have seen in the Introduction, each thread inside a process shares the same address space. This is a key point: it means that in C it is possible to implement threads with functions. Looking at fig. 3 we can see at line `87` that a thread is started by a call to `worker(arg)` where `worker` is in fig. 2, line `25`. A call to `pthread_create` immediately starts a thread `worker` and returns, so it is available to start other threads and the problems raise because all of the variables are inside the same address space, so we must be sure that each thread doesn't interfere with others threads. Now, in C variables are of three types (see Reese (2013, p. 2)):

static/global: A static variable has its scope inside the function where it is declared, but its lifetime is that of the process. A global variable has its scope inside the program and the same lifetime of the static variable. For this reason static and global variables need the special treatment assured by threads;

automatic: An automatic variable has its scope inside the function that declared it and its lifetime corresponds to the duration of the function. The C runtime *automatically* manages these variables so there are no problems with threads. Local variables and parameters of a function are automatic variables so they are *thread safe*;

dynamic: it is a variable created with `malloc` or equivalent function. This is usually a pointer that can be passed (by value) to a function, and hence is a sort of global access to the memory addressed by the pointer. Its lifetime ends when and whether the memory is freed. This means that it can lead to *memory leak* when the memory is never released. Threads must carefully manage a pointer of this type.

luaTEX has several global/static variables and a thread-safe version of the program implies a rewriting of a consistent part of the code; it is more convenient considering only the Lua interpreter because, as explained in Ierusalimschy (2013, p. 251), each Lua function receives a pointer to `lua_State` and uses exclusively this state, and this "implementation makes Lua reentrant and ready to be used in multithreaded code". We can hence follow the idea of Ierusalimschy (2013, sec. 31.2): each thread (the `worker` function) creates its own Lua state (in fig. 2, line `27`) and this interpreter executes a chunk of Lua code in (fig. 2, line `48`). As we can see in fig. 3, there are two dynamic variables: `code_clone` (line `70`), that has a copy of the chunk so there is no interference with the `lua_State` of luaTEX, and `arg` (line `75`) which is `worker` argument. It's not possible to free these variables in `swiglib_pthread_create`, because this function can return *before* the `worker` is started, so this last one would not see valid data. This means that `worker` has to free these variables after their use.

We can see an example in fig. 4, where three threads are created and all of them run the code

in code_base (line 17). As expected, the output in fig. 5 shows the printed lines output without a particular order. The thread workers are *joinable*, thus the master can synchronize itself to their termination (lines 46 to 48 in fig 4).

We can make the following considerations:

1. in the C code there is no use of any primitive to synchronize the Pthreads API (mutex, semaphore) because there are not any static/global variables and dynamic variables are not shared (they are managed by the worker). In spite of this, **the code is not thread-safe** due the void pointer data in the struct thread_worker_arg (fig. 2, line 23);

2. it seems that also in the Lua code there are not any shared resources, but **this is not true**. Indeed in fig. 4, line 27 there is a call to os.date() which can call the system function localtime(); this one is not thread-safe (see http://pubs.opengroup.org/onlinepubs/009695399/, subsec. 2.9.1) There is a thread-safe version of localtime, localtime_r, used by Lua 5.2.2 (and hence LuaTeX 0.76.0) if it is available at compile-time (i.e., in a POSIX system, but it works also under Linux even if it's not fully POSIX-compliant). It is *not used* in Lua 5.1.4 (and hence LuaTeX until TeX Live 2012). luatex.exe for Windows is linked against msvcrt.dll, *which is thread safe.*

Considering observation 1. we can say that the void pointer data is a way to pass user data to a thread. We show it with an example when the user data is an int. First we have to create a pointer to our data; this is done in the interface with the line %pointer_functions(int , int_p); (fig. 2, line 15). This means that on the Lua side there are now those functions to manage this type of user data:

```
local data = pthread.new_int_p()
pthread.int_p_assign(data,99999)
```

We can pass data to the thread:

```
pthread.swiglib_pthread_create(
   t0,attr,-1,
   code0,string.len(code0),
   data)
```

In fig. 2, lines 42 to 47 data is stored as a *lightuserdata* in the swiglib_pthread_data Lua variable and we can read it as shown in fig. 1.

The main advantage of the lightuserdata is that it is not managed by the garbage collector of Lua in the thread, so there is no risk for the pointed memory to be erased, but on the other side it is the caller's responsibility to protect and manage this resource because the worker doesn't take any action. In this sense worker is not thread-safe. Moreover, we must supply two other functions: pthread.lightuserdata_touserdata_void_p and pthread.void_p_to_int_p (in fig. 2, lines 91 and 108) to translate a lightuserdata to an int. If we don't need to pass data we can set data=nil as in fig. 4 and there is no risk because swiglib_pthread_data is also set to nil. This technique is general, because with SWIG we can define our data as a struct (as in fig. 2, line 20) and then provide the pointer functions and the lightuserdata conversion functions.

According to point 2., we can mitigate the problem selecting the module to load. The auxlibs of arg (which is passed by value, thus it's thread-safe) it's a bit mask to select which module to load into the Lua state of the thread (in fig. 2, lines 41 to 43): -1 loads the entire module. But in general we should check that the system functions used by Lua are thread-safe.

As we can see, Pthreads is a low-level library and its use requires some knowledge that is probably not so common to the average TeX user. In the next subsection we will see another library do the deal with concurrency in a more abstract way.

3.2 A SWIG wrapper for ZeroMQ

ZeroMQ is a C library for Message passing. In a broader sense, message passing concerns the exchange of messages between threads, be they in the same process, in different processes on the same instance of an OS, in different OSs on different machines. The key concept is a generalization of the UNIX© socket, the *zmq socket*, which has several *types*, each one defining a *message pattern*:

Request-reply: connects a set of clients to a set of services.

Publish-subscribe: connects a set of publishers to a set of subscribers. This is a data distribution pattern, which means that the reference data is published in a service bus and only new subscribers need to be added to the service bus; there is no need to change the service to serve new targets.

Push-pull: connects nodes in a fan-out / fan-in pattern that can have multiple steps and loops. This is a parallel task distribution and collection pattern.

Exclusive pair: connects two sockets in an exclusive pair. This is a low-level pattern for specific, advanced use cases and still experimental.

(please refer to http://en.wikipedia.org/wiki/Messaging_pattern.)

ZeroMQ is a mid-level abstraction library; it can be used to replace some functionality of Pthreads; it has not its own low-level mechanisms of synchronization but, on the other side, it has not any built-in server program (like http or ftp servers) — it can be used to build custom versions of these programs. For example, it is currently used by

```
1  local code_base =
2  [=[
3    local th = "%s"
4    local _core= package.loadlib("./core.so","luaopen_core")
5    if not(_core) then print(th .. ":error loading _core") return 1 end
6    local pthread = _core()
7    if swiglib_pthread_data~=nil then
8      local d1 = pthread.lightuserdata_touserdata_void_p(swiglib_pthread_data)
9      print(th.." data value=",pthread.int_p_value( pthread.void_p_to_int_p(d1) ) )
10   end
11   print(th .. " end")
12 ]=]
```

FIGURE 1: Passing a user data to a thread (no thread safe)

CERN to update their Controls Middleware system software previously based on the CORBA middleware (DWORAK *et al.*, 2012)[4]. The authoritative source of documentation is the site `http://zguide.zeromq.org` and the new book HINTJENS (2013).

A SWIG wrapper for ZeroMQ is straight:

```
%module core
%{
#include "zeromq/zmq.h"
%}
%include "zeromq/zmq_utils.h" ;
%include "zeromq/zmq.h" ;
```

Building the wrapper is also simple (here shown for Linux), and it needs the Pthreads library:

```
LUAINC52=/usr/include/lua5.2
LIBS="-lpthread -lzmq"
CFLAGS="-g  -O2 -Wall -I./zeromq \
           -pthread"
swig  -lua core.i
rm -vf core_wrap.o
gcc   -O2 -fpic -I./zeromq -I$LUAINC52 \
      -c core_wrap.c  -o core_wrap.o
rm -vf core.so
gcc -Wall -shared -O2 \
    -Wl,-rpath,'$ORIGIN/.'\
    $CFLAGS \
    core_wrap.o  \
    -llua5.2 $LIBS \
    -o core.so
```

In fig. 6 we show a push-pull example: the main process opens a `ZMQ_PULL` zmq socket connected to the TCP port 5555 and *launches* `max_workers` *processes in background* (line 20). After that it "pulls" the results with a `receive` (fig. 6, line 32). Lines 24 to 49 ensure that all workers are served[5]. Each worker shares the same file shown in fig. 7: it opens a `ZMQ_PUSH` zmq socket connected to the same TCP port and "pushes" data with a `send` (fig. 7, line 24). The mentioned data is an array of char, managed by the external library `helpers` — a simple SWIG wrapper to `char[]`:

```
%module core
%include "carrays.i"
%array_functions(char, char_array);
```

The key point here is that we don't use threads but processes, so there is no need for synchronized accesses; on the other side the way we run a process in background now depends on the underlying OS: under Windows we have to use the following string for `os.execute`:

```
"start 'worker' /b luatex.exe
  'ex008-PUSH_PULL-worker.lua' '%s' 2>&1"
```

ZeroMQ also manages message passing between threads with the protocol `inproc://<name>`. Fig. 8 and fig. 9 show the same push-pull pattern with threads. Running times are as expected: the process version lasts

```
$ time ./luatex ex008-PUSH_PULL-process.lua \
   >/dev/null
real 0m12.120s
user 0m0.196s
sys 0m0.852s
```

while the threaded version lasts

```
$ time ./luatex ex008-PUSH_PULL-joined.lua \
   >/dev/null
real 0m0.712s
user 0m0.064s
sys 0m4.952s
```

4. "We are migrating an infrastructure with 3500 active servers and 1500 active clients from CORBA to ZMQ. We only provide the library, our users implement the clients/servers on top of it and deploy where/how they want. We also have many combinations of hardware/OS: low level front-ends, middle tier servers and workstations." (see `http://comments.gmane.org/gmane.network.zeromq.devel/18437`).

5. It's a very primitive mechanism for synchronization: production code must use more robust solutions as explained in HINTJENS (2013).

i.e. approximately 17 faster. The approach of ZeroMQ is clear: *don't share state* and hence there is no need for synchronization to access shared resource. In fact the only object shared is the `context` which is declared as thread safe (on the other hand, a zmq socket is not thread safe so it must not be shared).

3.3 Considerations for integration in LuaTEX

The current implementations of LuaTEX, pdfTEX and XeTEX support at least Linux, OS X and Windows, so it's almost a mandatory requirement that an extension library also works at least under the same OSs. We have made our experiments on Linux 64bit and Windows Home premium 32bit and we have still to do tests under Windows 64bit. Due to the lack of a machine with the latest OS X, we have not made tests with this OS, but we are confident that there we would not have serious problems because it is a POSIX system and the clang compiler is a mature project, at least for the C language. An important problem is where to put these so/dll libraries without cluttering the OS: for example, under Linux, passing `-Wl,-rpath,'$ORIGIN/.'` to the linker has as a consequence that the libraries can be found starting from the directory of the local application, but a similar switch is not available under Windows which has different rules. LuaTEX has the Lua file system module `lfs` from `http://keplerproject.github.io/luafilesystem/` but it's not usable with a worker because it has a separated state. A solution is then to load a `lfs` module into the worker, so we can switch from the current directory to that one having the module to load, as in the following chunk:

```
_core = package.loadlib("lfs/lfs.dll",
                        "luaopen_lfs")
if not(_core) then print("error lfs")
  return 1 end
local lfs = _core()
local current_dir = lfs.currentdir()
lfs.chdir("./zeromq")
local _core= package.loadlib(
               "./zeromq/core.dll",
               "luaopen_core")
if not(_core) then print("error zmq")
  return 1 end
local zmq = _core()
lfs.chdir(current_dir)
```

The Swiglib project `http://www.luatex.org/swiglib.html` tries to address these and other aspects, as for example a generic `helper` module.

While it is clear that Pthreads and ZeroMQ open new possibilities to efficiently manage different data sources for typesetting, the other aspect — probably the most interesting one — of interacting with LuaTEX to add concurrency to the task of typesetting still remains to explore, but we can delineate two possibilities:

process: run several instances of LuaTEX coordinate by ZeroMQ, typically with a `tcp` connection. This could be useful for example in processing a book if some chapters are mutually independent;

thread: inside a single instance of LuaTEX use worker threads that cooperate with the Lua state of LuaTEX. For example a two columns layout could be rendered in a pipeline by two different threads, or n threads could be used in parallel to build a vbox with different parameters and then choose the best one.

The process possibility means that we should have identical instances, but this is usually not a problem (just uses the same version of LuaTEX and format for each instance); of course the runs cannot put files in the same folder because each instance would interfere with the others. The problem is to resolve the implicit dependencies (as, for example, page and chapter numbers), and the synchronization of the common parts, as for example the index.

The thread possibility looks more complex. Communicating with the Lua internal state of LuaTEX is complicated because there is not any API — which is reasonable: it is a program. A more feasible approach is a pure Lua implementation of some parts of LuaTEX, as for example the line breaking algorithm. In this case the problem is how to exchange data: `void *data` is not thread-safe, so it must be carefully managed.

4 Conclusions

The implicit parallelism offered by libraries like OpenMP can give LuaTEX an effective support for multiprocessors, but simple estimations indicate that the amount of code to rewrite is extremely huge so that any potential advantage is lost.

Explicit parallelism/concurrency by external libraries is more convenient for an existing program like LuaTEX that can take advantage of the very flexible Lua interpreter. The Pthreads and ZeroMQ libraries cover the "dual" aspect of processes vs threads giving a good coverage of the subject and we have shown some simple examples working with the Lua interpreter of LuaTEX. We think that these examples justify more investigations explicitly focused on typesetting problems, especially in the area of adding parallelism to LuaTEX using threads. We are open to suggestions and critiques to get the best solution.

The code is hosted in the SVN server of the Swiglib project at `https://foundry.supelec.fr/scm/viewvc.php/trunk/experimental/?root=swiglib`.

References

Breshears, C. (2009). *The Art of Concurrency: A Thread Monkey's Guide to Writing Parallel Applications*. O'Reilly Media, Inc.

Chapman, B., Jost, G. e Pas, R. v. d. (2007). *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*. The MIT Press.

Dworak, A., Ehm, F., Charrue, P. e Sliwinski, W. (2012). «The new CERN Controls Middleware». *Journal of Physics: Conference Series*, **396** (1), p. 012 017. URL `http://stacks.iop.org/1742-6596/396/i=1/a=012017`.

Hennessy, J. L. e Patterson, D. A. (2012). *Computer Architecture — A Quantitative Approach*. Morgan Kaufmann, 5ª edizione.

Hintjens, P. (2013). *Code Connected Volume 1: Learning ZeroMQ*. Code Connected Series. CreateSpace Independent Publishing Platform. URL `http://books.google.it/books?id=hY6olQEACAAJ`.

Ierusalimschy, R. (2013). *Programming in Lua, Third Edition*. Lua.Org.

Kerrisk, M. (2010). *The Linux Programming Interface: A Linux and UNIX System Programming Handbook*. No Starch Press, San Francisco, CA, USA, 1ª edizione.

Reese, R. (2013). *Understanding and Using C Pointers*. O'Reilly Media, Incorporated. URL `http://books.google.it/books?id=7dOwkQEACAAJ`.

Russinovich, M. E., Solomon, D. A. e Ionescu, A. (2012a). *Windows Internals, Part 1: Covering Windows Server 2008 R2 and Windows 7*. Microsoft Press, 6ª edizione.

— (2012b). *Windows Internals, Part 2: Covering Windows Server 2008 R2 and Windows 7 (Windows Internals)*. Microsoft Press.

▷ Luigi Scarso
`luigi dot scarso at gmail dot com`

```
%module core
%include "carrays.i"; %include "cpointer.i"; %include "constraints.i";
%include "cmalloc.i"; %include "lua_fnptr.i";
%{
#include "pthread/sched.h"
#include "pthread/semaphore.h"
#include "pthread/pthread.h"
%}
%ignore worker; %ignore thread_worker_arg;
%include "pthread/sched.h";
%include "pthread/semaphore.h"
%include "pthread/pthread.h"
%pointer_functions(pthread_attr_t,pthread_attr_t_p);
%pointer_functions(pthread_t,pthread_t_p);
%pointer_functions(int , int_p);
%array_functions(pthread_attr_t *,pthread_attr_t_p_array);
%array_functions(void *,void_p_array);
%pointer_cast(void *,int *,void_p_to_int_p);
%pointer_cast(int *,void *,int_p_to_void_p);
%inline %{
struct thread_worker_arg{
  char *code;          int auxlibs;
  size_t is_joinable;  void *data;
};
void *worker(void *thread_arg){
  struct thread_worker_arg *arg = (struct thread_worker_arg *) thread_arg;
  lua_State *L = luaL_newstate();
  char *code = arg->code; int auxlibs = arg->auxlibs;
  int res ;  int *retval;
  if (auxlibs<0){
    luaL_openlibs(L);
  } else if (auxlibs==0) {
    luaopen_base(L);
  } else {
    luaopen_base(L);
    if (auxlibs & (1<<0)) luaopen_coroutine(L);  if (auxlibs & (1<<1)) luaopen_table(L);
    if (auxlibs & (1<<2)) luaopen_io(L);     if (auxlibs & (1<<3)) luaopen_os(L);
    if (auxlibs & (1<<4)) luaopen_string(L); if (auxlibs & (1<<5)) luaopen_bit32(L);
    if (auxlibs & (1<<6)) luaopen_math(L);   if (auxlibs & (1<<7)) luaopen_debug(L);
    if (auxlibs & (1<<8)) luaopen_package(L);
  }
  if (arg->data==NULL) {
    lua_pushnil(L);
  } else {
    lua_pushlightuserdata (L, arg->data);
  }
  lua_setglobal(L, "swiglib_pthread_data");
  res = luaL_loadstring(L,code);
  if (res==0)
    res = lua_pcall(L,0,0,0);
  /* is this th joinable ?*/
  if (arg->is_joinable==PTHREAD_CREATE_JOINABLE){
    /* main thread must free retval */
    retval = malloc(sizeof(int));
    *retval = res;
  } else {
    retval=NULL;
  }
  free(arg->code);  free(arg);  lua_close(L);
  return (void *) retval;
}
%}
```

FIGURE 2: Interface file `core.i` (1 of 2)

```
#include <string.h>
int swiglib_pthread_create(pthread_t *thread,const pthread_attr_t *attr,
                           int auxlibs, const char *code, int code_len,
                           void *data) {
    int res; char *code_clone; int attr_value;
    if (code_len<0)  return -1000;
    if (code_len==0) return -1001;
    code_clone = malloc((sizeof(char)*code_len)+1);
    if (code_clone==NULL) return -1002;
    code_clone[code_len]=0;
    code_clone = strncpy(code_clone,code,code_len);
    if (code_clone==NULL) return -1003;
    struct thread_worker_arg *arg = malloc(sizeof(struct thread_worker_arg));/* NO ANSI C !*/
    if (arg==NULL) return -1005;
    /*worker(s) must free them !! */
    arg->code = code_clone;
    arg->auxlibs=auxlibs;
    arg->data = data;
    if (attr==NULL){
      arg->is_joinable=PTHREAD_CREATE_JOINABLE;
    } else {
      if (pthread_attr_getdetachstate(attr, &attr_value)!=0) return -1006;
      arg->is_joinable=attr_value;
    }
    res=pthread_create(thread, attr, worker, arg);
    return res;
}
%}
%native(lightuserdata_touserdata_int_p)
  static int native_lightuserdata_touserdata_int_p(lua_State*L);
%{int native_lightuserdata_touserdata_int_p(lua_State*L){
  int SWIG_arg = 0;
  int *result = 0 ;
  SWIG_check_num_args("lightuserdata_touserdata_int_p",1,1);
  if(!lua_islightuserdata(L,1))
    SWIG_fail_arg("lightuserdata_touserdata_int_p",1,"lightuserdata int *");
  result = (int *)lua_touserdata(L,1);
  SWIG_NewPointerObj(L,result,SWIGTYPE_p_int,0); SWIG_arg++;
  return SWIG_arg;
  if(0) SWIG_fail;
fail:
  lua_error(L);
  return SWIG_arg;
}
%}
%native(lightuserdata_touserdata_void_p)
   static int native_lightuserdata_touserdata_void_p(lua_State*L);
%{int native_lightuserdata_touserdata_void_p(lua_State*L){
  int SWIG_arg = 0;
  void *result = 0 ;
  SWIG_check_num_args("lightuserdata_touserdata_void_p",1,1);
  if(!lua_islightuserdata(L,1))
   SWIG_fail_arg("lightuserdata_touserdata_void_p",1,"lightuserdata void *");
  result = (void *)lua_touserdata(L,1);
  SWIG_NewPointerObj(L,result,SWIGTYPE_p_void,0); SWIG_arg++;
  return SWIG_arg;
  if(0) SWIG_fail;
fail:
  lua_error(L);
  return SWIG_arg;
}
%}
```

FIGURE 3: Interface file `core.i` (2 of 2)

```
local _core= package.loadlib("./core.so","luaopen_core")
if not(_core) then print("error loading _core") os.exit(1) end
local clock = os.clock
function sleep(n)  -- seconds
  local t0 = clock()
  while clock() - t0 <= n do end
end
local pthread = _core()
attr  = pthread.new_pthread_attr_t_p()
pthread.pthread_attr_init(attr)
pthread.pthread_attr_setdetachstate(attr, pthread.PTHREAD_CREATE_JOINABLE)

t0    = pthread.new_pthread_t_p()
t1    = pthread.new_pthread_t_p()
t2    = pthread.new_pthread_t_p()

local code_base =
[=[
  local clock = os.clock
  function sleep(n)  -- seconds
   local t0 = clock()
   while clock() - t0 <= n do end
  end
  local s,s1
  local th="%s"
  for i=1,%s do
   local s=os.date()
   local s1=th.."<"..tostring(s).." "..tostring(i)..">"
   io.write(s1,"\n")
   sleep(%s)
  end
]=]


local code0=string.format(code_base,"1","10","0.4")
local code1=string.format(code_base,"2","10","0.3")
local code2=string.format(code_base,"3","10","0.4")
local data = nil
local rc

rc = pthread.swiglib_pthread_create(t0,attr,-1,code0,string.len(code0),nil)
rc = pthread.swiglib_pthread_create(t1,attr,-1,code1,string.len(code1),nil)
rc = pthread.swiglib_pthread_create(t2,attr,-1,code2,string.len(code2),nil)
pthread.pthread_attr_destroy(attr);

pthread.pthread_join(pthread.pthread_t_p_value(t0),nil)
pthread.pthread_join(pthread.pthread_t_p_value(t1),nil)
pthread.pthread_join(pthread.pthread_t_p_value(t2),nil)

print("end")
```

FIGURE 4: Test running 3 threads joined (1 of 2)

```
2<Fri Sep 13 18:07:54 2013 1>
3<Fri Sep 13 18:07:54 2013 1>
1<Fri Sep 13 18:07:54 2013 1>
2<Fri Sep 13 18:07:54 2013 2>
3<Fri Sep 13 18:07:54 2013 2>
1<Fri Sep 13 18:07:54 2013 2>
2<Fri Sep 13 18:07:54 2013 3>
3<Fri Sep 13 18:07:54 2013 3>1<Fri Sep 13 18:07:54 2013 3>

2<Fri Sep 13 18:07:54 2013 4>
1<Fri Sep 13 18:07:54 2013 4>
3<Fri Sep 13 18:07:54 2013 4>
2<Fri Sep 13 18:07:54 2013 5>
2<Fri Sep 13 18:07:54 2013 6>
3<Fri Sep 13 18:07:54 2013 5>
1<Fri Sep 13 18:07:54 2013 5>
2<Fri Sep 13 18:07:54 2013 7>
3<Fri Sep 13 18:07:54 2013 6>
1<Fri Sep 13 18:07:54 2013 6>
2<Fri Sep 13 18:07:55 2013 8>
3<Fri Sep 13 18:07:55 2013 7>
1<Fri Sep 13 18:07:55 2013 7>
2<Fri Sep 13 18:07:55 2013 9>
2<Fri Sep 13 18:07:55 2013 10>
3<Fri Sep 13 18:07:55 2013 8>
1<Fri Sep 13 18:07:55 2013 8>
3<Fri Sep 13 18:07:55 2013 9>
1<Fri Sep 13 18:07:55 2013 9>
3<Fri Sep 13 18:07:55 2013 10>
1<Fri Sep 13 18:07:55 2013 10>
end
################################################################
3<Fri Sep 13 18:07:55 2013 1>
1<Fri Sep 13 18:07:55 2013 1>
2<Fri Sep 13 18:07:55 2013 1>
2<Fri Sep 13 18:07:55 2013 2>
3<Fri Sep 13 18:07:55 2013 2>
1<Fri Sep 13 18:07:55 2013 2>
2<Fri Sep 13 18:07:56 2013 3>
1<Fri Sep 13 18:07:56 2013 3>
3<Fri Sep 13 18:07:56 2013 3>
2<Fri Sep 13 18:07:56 2013 4>
1<Fri Sep 13 18:07:56 2013 4>
3<Fri Sep 13 18:07:56 2013 4>
2<Fri Sep 13 18:07:56 2013 5>
2<Fri Sep 13 18:07:56 2013 6>
1<Fri Sep 13 18:07:56 2013 5>
3<Fri Sep 13 18:07:56 2013 5>
2<Fri Sep 13 18:07:56 2013 7>
3<Fri Sep 13 18:07:56 2013 6>1<Fri Sep 13 18:07:56 2013 6>

2<Fri Sep 13 18:07:56 2013 8>
1<Fri Sep 13 18:07:56 2013 7>
3<Fri Sep 13 18:07:56 2013 7>
2<Fri Sep 13 18:07:56 2013 9>
2<Fri Sep 13 18:07:56 2013 10>
3<Fri Sep 13 18:07:56 2013 8>
1<Fri Sep 13 18:07:56 2013 8>
1<Fri Sep 13 18:07:56 2013 9>
3<Fri Sep 13 18:07:56 2013 9>
3<Fri Sep 13 18:07:57 2013 10>
1<Fri Sep 13 18:07:57 2013 10>
end
```

FIGURE 5: Test running 3 threads joined, two outputs (2 of 2)

```
print("PULL BEGIN "..os.date())
local _core= package.loadlib("./zeromq/core.so","luaopen_core")
if not(_core) then print(th .. ":error zmq") return 1 end
local zmq = _core()
local f = package.loadlib("helpers/core.so","luaopen_core")
if not(f) then print("Error on loading helpers" ) return 1 end
local helpers = f()
local clock = os.clock
function sleep(n)  -- seconds
  local t0 = clock()
  while clock() - t0 <= n do end
end
local max_workers = 10
-- Socket to receive  messages on
local context  = zmq.zmq_ctx_new();
local recvfrom = zmq.zmq_socket(context, zmq.ZMQ_PULL);
local rc       = zmq.zmq_bind(recvfrom, "tcp://*:5555");
if (rc~=0) then print("error on bind tcp://workers") return 1 end
for k=1,max_workers do
 os.execute(string.format("lua5.2 'ex008-PUSH_PULL-worker.lua' '%s'  2>&1 &",k))
end
local c={}
local i=max_workers+3 -- uhu a sign that push-pull is  wrong
while (i>0) do
  local size = -1
  local bufsize = 256
  local _buffer     = helpers.new_char_array(bufsize)
  helpers.char_array_setitem(_buffer,bufsize-1,"\0")
  local buffer = helpers.char_p_to_void_p(_buffer)
  --msg_size doesn't include the trailing zero "\0"
  msg_size=bufsize-1
  size = zmq.zmq_recv(recvfrom, buffer, msg_size,0);
  local data=""
  if size==-1 then
    data=""
  else
    if size >(bufsize)  then
      size=bufsize-1
    end
    buffer = helpers.void_p_to_char_p(buffer)
    helpers.char_array_setitem(buffer,size,"\0")
    local t={}
    for i=0,size-1 do t[#t+1]=helpers.char_array_getitem(buffer,i) end
    data=table.concat(t)
  end
  c[#c+1]=string.format("SINK: seen %s size=%s data=%s,#data=%s",i-3,size,data,#data)
  i=i-1
  if i==3 then sleep(1) ; i=0 end
end
for k=1,#c do print(c[k]) end
zmq.zmq_close(recvfrom);
zmq.zmq_ctx_destroy(context);
print("PULL END "..os.date())
```

Figure 6: Push-pull: the code of the pull process (1 of 2)

```
local th = tostring(arg[1])
local f = package.loadlib("helpers/core.so","luaopen_core")
if not(f) then print("Error on loading helpers" ) return 1 end
local helpers = f()
local _core= package.loadlib("./zeromq/core.so","luaopen_core")
if not(_core) then print(th .. ":error loading _core") return 1 end
local zmq = _core()
local clock = os.clock
function sleep(n)  -- seconds
 local t0 = clock()
 while clock() - t0 <= n do end
end
local context = zmq.zmq_ctx_new();
local sendto = zmq.zmq_socket(context, zmq.ZMQ_PUSH);
local rc     = zmq.zmq_connect(sendto, "tcp://localhost:5555");
if (rc~=0) then print(th.." worker:error on connect tcp://localhost") return 1 end
local msg = "123456_"..th
local bufsize= #msg+1
local zmq_msg= helpers.new_char_array(bufsize)
helpers.char_array_setitem(zmq_msg,bufsize-1,"\0")
for i=1,#msg do
  helpers.char_array_setitem(zmq_msg,i-1,string.char(string.byte(msg,i)))
end
local res = zmq.zmq_send(sendto, helpers.char_p_to_void_p(zmq_msg), #msg, 0);
zmq.zmq_close(sendto);
zmq.zmq_ctx_destroy(context);
sleep(1)
```

FIGURE 7: Push-pull: the code of a push worker (2 of 2)

```
print("BEGIN PULL"..os.date())
local _core= package.loadlib("./core.so","luaopen_core")
if not(_core) then print("error pthread") os.exit(1) end
local pthread = _core()
local _core= package.loadlib("./zeromq/core.so","luaopen_core")
if not(_core) then print(th .. ":error zmq") return 1 end
local zmq = _core()
local f = package.loadlib("helpers/core.so","luaopen_core")
if not(f) then print("Error on loading helpers" ) return 1 end
local helpers = f()
local clock = os.clock
function sleep(n)  -- seconds
  local t0 = clock()
  while clock() - t0 <= n do end
end
local attr  = pthread.new_pthread_attr_t_p()
local max_workers = 100 --512
local limit_socket = 512 -- FD_SIZELIMIT  ?
if max_workers>limit_socket then max_workers=limit_socket end
local th = {}
for k=1,max_workers do th[k]=pthread.new_pthread_t_p() end
pthread.pthread_attr_init(attr)
pthread.pthread_attr_setdetachstate(attr, pthread.PTHREAD_CREATE_JOINABLE)
local data = nil
local worker_skeleton =
[=[
  local th = "%s"
  local f = package.loadlib("helpers/core.so","luaopen_core")
  if not(f) then print("Error on loading helpers" ) return 1 end
  local helpers = f()
  local _core= package.loadlib("./zeromq/core.so","luaopen_core")
  if not(_core) then print(th .. ":error loading _core") return 1 end
  local zmq = _core()
  local clock = os.clock
  function sleep(n)  -- seconds
   local t0 = clock()
   while clock() - t0 <= n do end
  end
  if swiglib_pthread_data == nil then print("error with swiglib_pthread_data") return 1 end
  -- Socket to send messages to
  local context = zmq.lightuserdata_touserdata_void_p(swiglib_pthread_data)
  local sendto = zmq.zmq_socket(context, zmq.ZMQ_PUSH);
  local rc     = zmq.zmq_connect(sendto, "inproc://workers");
  if (rc~=0) then print(th.." worker:error on connect inproc://workers") return 1 end
  local msg = "123456_"..th
  local bufsize= #msg+1
  local zmq_msg= helpers.new_char_array(bufsize)
  helpers.char_array_setitem(zmq_msg,bufsize-1,"\0")
  for i=1,#msg do
    helpers.char_array_setitem(zmq_msg,i-1,string.char(string.byte(msg,i)))
  end
  local res = zmq.zmq_send(sendto, helpers.char_p_to_void_p(zmq_msg), #msg, 0);
  zmq.zmq_close(sendto);
  sleep(1)
]=]
local worker={}
for k=1,max_workers do worker[k]=string.format(worker_skeleton,tostring(k)) end
```

FIGURE 8: Push-pull with threads (1 of 2)

```
-- Socket to receive  messages on
local context  = zmq.zmq_ctx_new();
local recvfrom = zmq.zmq_socket(context, zmq.ZMQ_PULL);
local rc       = zmq.zmq_bind(recvfrom, "inproc://workers");
if (rc~=0) then print("error on bind inproc://workers") return 1 end
data = context
for k=1,max_workers do
 pthread.swiglib_pthread_create(th[k],attr,-1,worker[k],string.len(worker[k]),data)
end
local c={}
local i=max_workers+3 -- uhu a sign of bad design in  push-pull
while (i>0) do
  local size = -1
  local bufsize = 256
  local _buffer     = helpers.new_char_array(bufsize)
  helpers.char_array_setitem(_buffer,bufsize-1,"\0")
  local buffer = helpers.char_p_to_void_p(_buffer)
  --msg_size doesn't include the trailing zero "\0"
  msg_size=bufsize-1
  size = zmq.zmq_recv(recvfrom, buffer, msg_size,0);
  local data=""
  if size==-1 then
    data=""
  else
    if size >(bufsize)  then
      size=bufsize-1
    end
    buffer = helpers.void_p_to_char_p(buffer)
    helpers.char_array_setitem(buffer,size,"\0")
    local t={}
    for i=0,size-1 do t[#t+1]=helpers.char_array_getitem(buffer,i) end
    data=table.concat(t)
  end
  c[#c+1]=string.format("SINK: seen %s size=%s data=%s,#data=%s",i-3,size,data,#data)
  i=i-1
  if i==3 then sleep(1) ; i=0 end
end
for k=1,#c do print(c[k]) end
zmq.zmq_close(recvfrom);
zmq.zmq_ctx_destroy(context);
print("END PULL"..os.date())
```

FIGURE 9: Push-pull with threads (2 of 2)

Sommari (indici generali) personalizzati

Gianluca Pignalberi

Sommario

Capita spesso che l'aspetto del sommario, o indice generale, standard di LaTeX non corrisponda a quanto ci serve. Sebbene siamo liberi di riprogrammare le parti di codice adibite alla composizione del sommario, usare il pacchetto titletoc può facilitare il compito anche e soprattutto a chi non è a proprio agio con la programmazione in TeX. Quattro casi d'uso reali mostreranno quanto proposto.

Abstract

It often happens that the LaTeX's standard look for tables of contents does not match what we need. Though we are free to reprogram those code lines that typeset ToCs, using the titletoc package may make our task easier, especially if we are not so skilled in TeX programming. Four actual cases show what we propose.

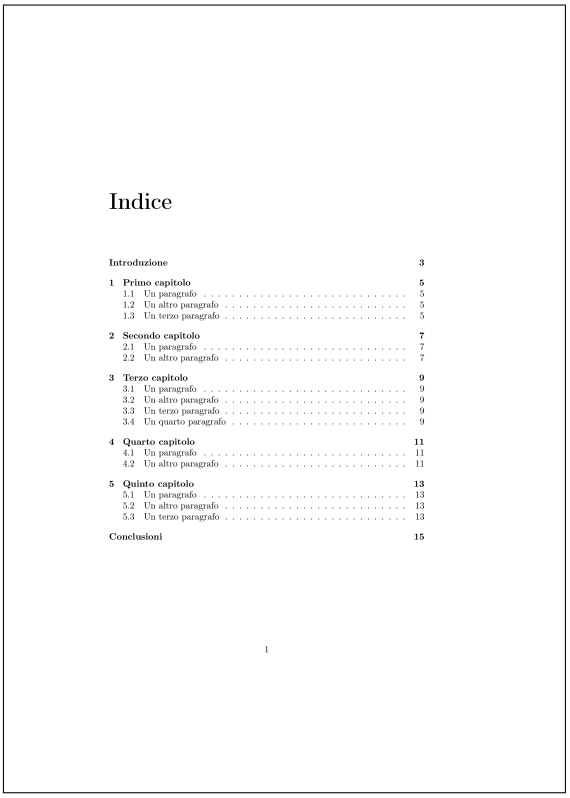
1 Introduzione

L'abitudine a usare LaTeX nella maniera più semplice possibile ci ha reso avvezzi a vedere i sommari (qui intesi nell'accezione di indici generali o *table of contents*) tutti dello stesso aspetto. Questo aspetto "standard" può essere soddisfacente in molti ambiti, ma può non andare bene se l'esigenza è quella di preparare un documento per un editore il cui stile è ben definito, riconoscibile e immutabile. È vero che l'editore che accetti manoscritti LaTeX è sicuramente dotato di classi e stili da distribuire ai propri autori, ma è pur vero che per alcuni editori accettare manoscritti LaTeX è un'eccezione che rende oneroso produrre quei file, e quindi il compito ricadrà sull'autore stesso o su un TeXnico ingaggiato all'uopo.

Nessuno ci impedisce di personalizzare l'aspetto di ogni parte del documento, ad esempio un libro, per riprodurre al meglio l'aspetto dei documenti di partenza. In questo breve articolo ci concentreremo sulla personalizzazione dell'aspetto dei sommari tramite il pacchetto titletoc.

2 Lo standard fissato da book.cls

Il tipico aspetto di un sommario generato da LaTeX è quello mostrato nella figura 1. Mi sono limitato a comporre un documento composto solo da capitoli e da paragrafi (`\section`). Se avessi messo dei sottoparagrafi (`\subsection`), avrei ottenuto le relative voci ulteriormente indentate: numero a tre livelli (x.x.x) allineato ai titoli dei paragrafi e il relativo titolo separato da uno spazio uguale a quello che separa dai numeri i capitoli e i paragrafi.


Indice

Introduzione 3

1 Primo capitolo 5
1.1 Un paragrafo . . . 5
1.2 Un altro paragrafo . . . 5
1.3 Un terzo paragrafo . . . 5

2 Secondo capitolo 7
2.1 Un paragrafo . . . 7
2.2 Un altro paragrafo . . . 7

3 Terzo capitolo 9
3.1 Un paragrafo . . . 9
3.2 Un altro paragrafo . . . 9
3.3 Un terzo paragrafo . . . 9
3.4 Un quarto paragrafo . . . 9

4 Quarto capitolo 11
4.1 Un paragrafo . . . 11
4.2 Un altro paragrafo . . . 11

5 Quinto capitolo 13
5.1 Un paragrafo . . . 13
5.2 Un altro paragrafo . . . 13
5.3 Un terzo paragrafo . . . 13

Conclusioni 15

1

FIGURA 1: Sommario dall'aspetto standard generato da LaTeX per un documento di classe *book*.

Ulteriori sottolivelli (quelli standard sono `\subsubsection`, `\paragraph`, `\subparagraph`) non vengono mostrati nel sommario, sebbene saranno presenti nel file `.toc`. Possiamo farli comparire facilmente copiando il comando giusto dal file book.cls e modificandolo opportunamente.[1] Il comando da modificare si trova alla riga 597 di book.cls versione 1.4h:

```
\setcounter{tocdepth}{2}
```

1. La pratica di modificare i file standard delle distribuzioni TeX è sconsigliata e peraltro vietata espressamente dalle versioni più vecchie della LaTeX Project Public License (THE LATEX PROJECT) per uso non personale. Ovviamente nessuno troverà niente da ridire se modifico book.cls per uso personale e non la ridistribuisco, ma per mantenere l'aspetto dell'immensa mole di documenti prodotta nel mondo e la coerenza dei pacchetti, la prassi era modificare una classe o uno stile solo dopo averli rinominati. L'ultima versione della LPPL permette anche le modifiche senza rinominare i file a patto di includere un chiaro rimando al file originale e una lista completa delle modifiche apportate.

indica fino a quale livello di titoli sarà mostrato nel sommario: `-1` mostra solo `\part`, `0` mostra anche `\chapter`, `1` aggiunge `\section` al sommario, `2` mostra fino a `\subsection`. Aumentare il numero di livello aumenta la quantità di informazione mostrata nell'indice.

Se aumentare la quantità di informazione da mostrare nell'indice è facilissimo, modificarne l'aspetto non lo è altrettanto perché bisogna modificare più o meno pesantemente il codice relativo alla composizione dell'indice stesso. Nel caso di book.cls, tale codice sta parte in `book.cls` e parte in `latex.ltx`. Chi abbia esperienza di programmazione di TEX non avrà difficoltà a copiare nel proprio file il codice opportuno e modificarlo per soddisfare le proprie esigenze. Chi invece non è in grado di svolgere almeno sufficientemente questa programmazione troverà comodo ricorrere a un pacchetto che facilita parte del compito: titletoc.

3 Il pacchetto titletoc

Nelle capaci "tasche" del CTAN e della TEX Live troviamo un insieme di pacchetti raggruppati sotto il nome collettivo di titlesec (Bezos, 2011). Questi pacchetti sono un rimpiazzo parziale o totale delle macro di LATEX relative ai titoli, alle testate e agli indici. Al suo interno, dunque, troviamo il pacchetto titletoc, quello predisposto alla modifica dell'aspetto dei sommari.

L'autore del pacchetto spiega nel manuale che titletoc ridefinisce totalmente i comandi per generare un sommario piuttosto che agganciarsi ai comandi definiti da LATEX. In questo modo è possibile personalizzare maggiormente l'aspetto dei sommari. Possiamo ricorrere soprattutto ad alcuni comandi di LATEX all'interno di quelli di titletoc così da avere ancora più libertà.

Anche il comportamento del pacchetto è differente rispetto allo standard di LATEX: le pagine del sommario possono essere interrotte solo tra due elementi dello stesso livello (per esempio, la pagina può essere interrotta tra due capitoli successivi) o tra due elementi di cui il primo è di un livello inferiore del secondo (per esempio, il primo è un paragrafo e il secondo è un capitolo). Non ci saranno invece interruzioni tra un elemento di livello superiore e uno di livello inferiore (ad esempio tra un capitolo e un paragrafo; in questo caso il paragrafo rimarrà nella stessa pagina del capitolo di appartenenza). Anche i simboli nelle righe "riempitive" non vengono centrati ma imbandierati a destra.

Gli elementi del sommario sono trattati come aree rettangolari contenenti del testo e, probabilmente, un riempitivo. Quest'area rettangolare può essere "deformata" aggiungendo o togliendo dello spazio all'inizio della prima e alla fine dell'ultima riga. Questo equivale a passare dalla situazione della figura 2 a quella della figura 3.

1.1 Il titolo di qugsto paragrafo è talmente lungo che una sola riga non basta a contenerlo e ne serve necessariamente una seconda e una terza 5

Figura 2: Blocco rettangolare: tutte le informazioni della voce dell'indice sono allineate al perimetro di un rettangolo ideale.

1.1 Il titolo di questo paragrafo è talmente lungo che una sola riga non basta a contenerlo e ne serve necessariamente una seconda e una terza 3

Figura 3: Blocco deformato: solo il titolo del paragrafo e il riempitivo rientrano in un rettangolo ideale; il numero di paragrafo e di pagina sono all'esterno, con uno spazio bianco sopra o sotto di essi.

Vediamo brevemente i comandi forniti dal pacchetto, così da avere un quadro d'insieme delle sue capacità e un'idea preliminare di come ottenere gli effetti visibili nelle figure 2 e 3.

Composizione delle voci

Abbiamo a disposizione due comandi per comporre le voci di un sommario:

`\dottedcontents` Compone una voce dell'indice completa di guida (la serie di puntini o altro simbolo che conduce l'occhio dal titolo al numero di pagina — *leader* —). La sua sintassi è:

```
\dottedcontents{<sezione>}
   [<sinistra>]
   {<precodice>}
   {<larghezza etichetta>}
   {<larghezza guida>}
```

`\titlecontents` Compone una voce dell'indice in maniera meno rigida del comando precedente. La sua sintassi è:

```
\titlecontents{<sezione>}
   [<sinistra>]
   {<precodice>}
   {<formato elemento numerato>}
   {<formato elemento non numerato>}
   {<formato riempitivo>}
   [<postcodice>]
```

Di questo comando esiste la versione *starred* che pone nello stesso capoverso, una dopo l'altra, tutte le voci d'indice del livello indicato in `<sezione>`.

Elementi delle voci

Le voci del comando `\dottedcontents` sono, nell'ordine, il nome della sezione in cui è strutturato il documento (chapter, section. . .), lo spazio (obbligatorio pur stando tra parentesi quadre) tra il margine sinistro e il testo, cioè il titolo del capitolo o paragrafo, l'eventuale codice da eseguire prima di comporre la voce dell'indice, la larghezza dell'etichetta, cioè lo spazio da lasciare per porre il

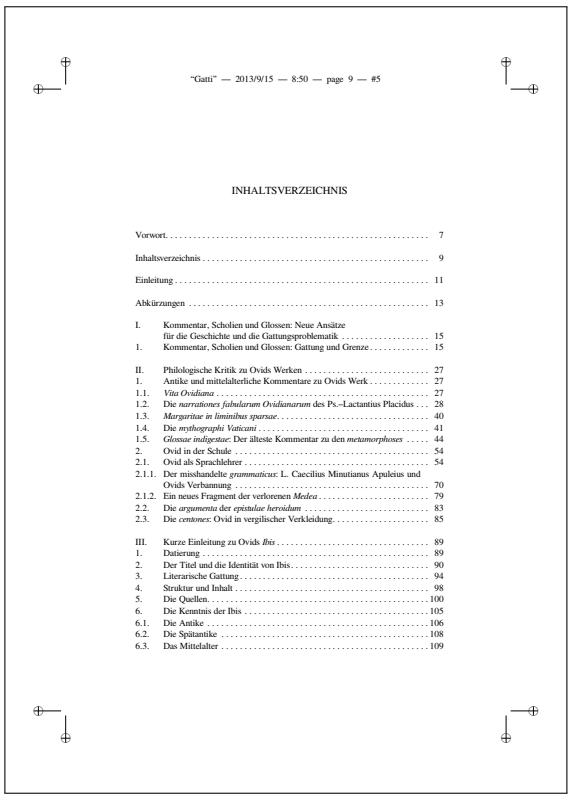


FIGURA 4: Riproduzione in LaTeX dell'indice di una collana dell'editore Franz Steiner Verlag.

numero di capitolo o paragrafo, e la larghezza della scatola contenente il carattere che compone la guida (il puntino).

Il seguente codice riproduce abbastanza fedelmente l'aspetto dell'indice generale di una collana dell'editore tedesco Franz Steiner Verlag, indice mostrato nella figura 4:

```
\dottedcontents{chapter}[3em]%
  {\addvspace{11.9pt}}{3em}{.5em}{}
\dottedcontents{section}[3em]%
  {}{3em}{.5em}{}
\dottedcontents{subsection}[3em]%
  {}{3em}{.5em}{}
\dottedcontents{subsubsection}[3em]%
  {}{3em}{.5em}{}
```

Tale codice lascia uno spazio di 3 em per contenere il numero di capitolo o paragrafo e fa iniziare i relativi titoli immediatamente dopo; la larghezza della scatola contenente il puntino della guida è sempre di 0.5 em. L'unico caso in cui è previsto un precodice è la spaziatura verticale di un capitolo dalla voce d'indice che lo precede. Naturalmente tutte le spaziature, orizzontali e verticali, dovranno essere adattate allo specifico documento per evitare suddivisioni incoerenti del sommario o sovrapposizioni tra il numero di un paragrafo e il relativo titolo (mentre 3 em è sufficiente a contenere il numero del sottoparagrafo 2.1.2, può non esserlo per contenere l'eventuale numero del sottoparagrafo 107.65.12).

I parametri da passare a `\titlecontents` invece sono, nell'ordine: il nome della sezione così come visto per `\dottedcontents`, lo spazio da lasciare tra il margine sinistro e il testo della voce dell'indice (anche qui obbligatorio pur stando tra parentesi quadre), l'eventuale codice da eseguire prima di comporre la voce d'indice, il formato delle voci numerate (in modalità orizzontale e usato immediatamente prima di comporre la voce d'indice; l'ultimo comando può prendere un argomento col titolo da scrivere nella voce d'indice), il formato delle voci non numerate (nella stessa modalità di quelle numerate) e, infine, l'eventuale codice da eseguire dopo aver composto la voce d'indice (per esempio una spaziatura verticale).

Correzione delle dimensioni

In alcuni casi lo spazio destinato a contenere il numero di pagina può non essere sufficiente. Il comando `\contentsmargin{<right>}` serve proprio ad aggiustare quello spazio per soddisfare ogni esigenza di composizione diversa dalla standard.

Le cose che non ti ho mai detto

È opportuno puntualizzare che il pacchetto titletoc presenta altri comandi che non sono stati riportati in questo articolo, oltre a offrire una sintassi ben più ricca per alcuni dei comandi qui segnalati.

Poiché lo scopo del presente articolo è quello di mostrare ai lettori come riprodurre facilmente dei sommari di aspetto eterogeneo e non di essere una guida all'uso di titletoc, non mi dilungherò ulteriormente in discussioni su quale comando fa cosa e di quali parametri necessita. Il manuale del pacchetto (Bezos, 2011) è stato scritto proprio per questo scopo.

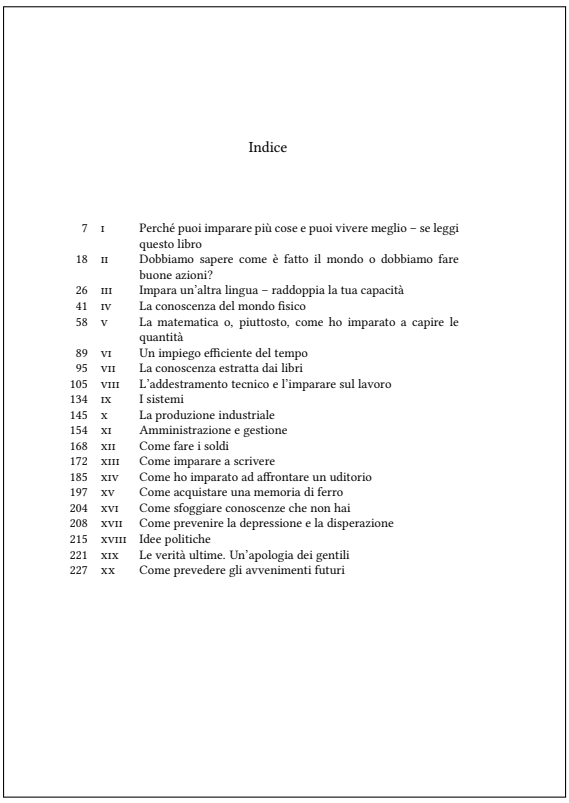
4 Personalizzare il sommario

È arrivato il momento di vedere alcuni esempi pratici della personalizzazione di un sommario con titletoc. Prendiamo tre esempi in rappresentanza dei più diffusi stili di composizione di un sommario. I tre esempi sono: Vacca (1987), Kernighan e Pike (1999) e Varian (2012).

4.1 Il sommario di *Come imparare...*

La figura 5 mostra la riproduzione del sommario di Vacca (1987) ottenuta con LaTeX e titletoc. Come per i paragrafi relativi agli altri due libri di riferimento, illustriamo l'indice per poi darne la descrizione con il codice titletoc.

Il titolo del capitolo (Indice) è centrato e posto al margine superiore della gabbia. Il font è di poco superiore a quello del testo, in tondo e alto/basso (iniziale maiuscola e il resto minuscolo). L'impostazione dello stile del titolo attiene alla classe usata e si può modificare direttamente agendo sulla classe o usando un pacchetto come titlesec (Bezos,



Indice

7 I Perché puoi imparare più cose e puoi vivere meglio – se leggi questo libro
18 II Dobbiamo sapere come è fatto il mondo o dobbiamo fare buone azioni?
26 III Impara un'altra lingua – raddoppia la tua capacità
41 IV La conoscenza del mondo fisico
58 V La matematica o, piuttosto, come ho imparato a capire le quantità
89 VI Un impiego efficiente del tempo
95 VII La conoscenza estratta dai libri
105 VIII L'addestramento tecnico e l'imparare sul lavoro
134 IX I sistemi
145 X La produzione industriale
154 XI Amministrazione e gestione
168 XII Come fare i soldi
172 XIII Come imparare a scrivere
185 XIV Come ho imparato ad affrontare un uditorio
197 XV Come acquistare una memoria di ferro
204 XVI Come sfoggiare conoscenze che non hai
208 XVII Come prevenire la depressione e la disperazione
215 XVIII Idee politiche
221 XIX Le verità ultime. Un'apologia dei gentili
227 XX Come prevedere gli avvenimenti futuri

Figura 5: Riproduzione in LaTeX dell'indice di Vacca (1987).

2011). Entrambe le pratiche esulano dall'obiettivo dell'articolo e non ne parleremo.

Ogni voce dell'indice contiene, da sinistra a destra:

- numero di pagina imbandierato a destra in un box di dimensioni fisse;
- numero di capitolo scritto in numeri romani maiuscoletti imbandierato a sinistra in un box di dimensioni fisse;
- titolo del capitolo giustificato in un box multiriga di larghezza pari al resto della riga.

Il libro non contiene paragrafi, pertanto definiremo i soli comandi dei capitoli, tutti numerati. Il codice per titletoc è il seguente:

```
1 \usepackage{titletoc}
2 \contentsmargin{0pt}
3 \titlecontents{chapter}[4pc]
4   {\contentsmargin{0pt}\makebox
      [0pt][r]{\thecontentspage
      \quad\makebox[30pt][l]{
      \textsc{\romannumeral
      \thecontentslabel\quad}}}}
5   {}
6   {}
7   {}
```

Nella riga 4 troviamo la definizione di quello che bisogna comporre prima di ogni capitolo, numerato o

7 I Perché puoi imparare più cose e puoi vivere meglio – se leggi questo libro

Figura 6: Indice di Vacca (1987) con i box separati.

meno. Il numero di pagina (`\thecontentspage`) è contenuto in un box con imbandieramento a destra. Nello stesso box componiamo un altro box, imbandierato a sinistra, contenente il numero di capitolo in numeri romani maiuscoletti. L'uso di due box uno dentro l'altro non è alternativo all'uso di due box separati. Se infatti sostituissimo la riga 4 con `\makebox[0pt][r]{\thecontentspage\quad} \makebox[30pt][l]{\textsc{\romannumeral \thecontentslabel\quad}}` otterremmo un risultato diverso: il titolo spezzato in due o più righe si troverebbe allineato al numero di capitolo, come vediamo nella figura 6.

Non serve riempire gli altri parametri del comando `\titlecontents` perché il lavoro è terminato.

4.2 Il sommario di *Programmazione...*

Il sommario di Kernighan e Pike (1999) (figura 7) è un po' più simile a quelli classici; cambia solo qualche decorazione e l'uso del grassetto.

La prima cosa a cambiare è il titolo, qui "Sommario" e non "Indice". Vediamo però la parte che ci compete. Le righe del sommario relative a un capitolo sono seguite da un filetto orizzontale a tutta larghezza. I titoli di ogni capitolo, numerato o no, sono in grassetto e così anche i relativi numeri di pagina e l'etichetta "Capitolo X". La riga del sommario corrispondente al capitolo numerato ha tre elementi: l'etichetta "Capitolo" con relativo numero imbandierati a sinistra in un box di dimensione fissa; il titolo del capitolo imbandierato a sinistra in un altro box; il numero di pagina imbandierato a destra alla fine della riga. Se invece il capitolo non è numerato (cioè *starred*) il titolo è imbandierato a sinistra senza alcuna etichetta. Vediamo il codice:

```
1 \titlecontents{chapter}[0pc]
2   {\addvspace{6pt}}
3   {\makebox[6em][l]{\textbf{
      Capitolo~\thecontentslabel
      }}\textbf}
4   {\textbf}
5   {\hfill\MakeTextUppercase{
      \textbf{\thecontentspage}}
      \\[-.75em]\rule{\textwidth
      }{.4pt}}
```

La riga 2 aggiunge uno spazio verticale per dividere la riga del capitolo da quanto la precede. La riga 3, quella relativa ai capitoli numerati, pone un box di larghezza fissa con l'etichetta "Capitolo", il numero e il titolo tutto in grassetto. La riga 4, quella dei capitoli starred, deve solo mettere

Sommario

Prefazione		**IX**
Capitolo 1	**Stile**	**1**
1.1	Nomi	3
1.2	Espressioni e istruzioni	6
1.3	Consistenza ed espressioni idiomatiche	11
1.4	Macro di funzioni	19
1.5	Numeri magici	21
1.6	Commenti	25
1.7	Perché preoccuparsi?	31
Capitolo 2	**Algoritmi e strutture dati**	**33**
2.1	Ricerca	34
2.2	Ordinamento	36
2.3	Librerie	39
2.4	Un quicksort Java	43
2.5	Notazione O	45
2.6	Crescita negli array	47
2.7	Liste	50
2.8	Alberi	56
2.9	Tabelle hash	62
2.10	Riepilogo	66
Capitolo 3	**Progetto e implementazione**	**69**
3.1	L'algoritmo della catena di Markov	70
3.2	Alternative alla struttura dati	72
3.3	Costruzione della struttura dati in C	74
3.4	Generazione dell'output	78
3.5	Java	81
3.6	C++	85
3.7	Awk e Perl	88
3.8	Prestazioni	90
3.9	Lezioni	92

FIGURA 7: Indice di KERNIGHAN e PIKE (1999) riprodotto con LaTeX.

in grassetto il titolo del capitolo. Infine la riga 5, quella relativa al riempitivo, pone tutto a destra il numero di pagina, quindi lascia uno spazio verticale e inserisce un filetto a tutta larghezza. Noterete che il numero di pagina è in maiuscolo. Se la cosa non ha senso per i numeri arabi, è obbligatoria invece per i numeri di pagina del *frontmatter*. Se non mettessimo il comando per il maiuscolo avremmo degli orrendi numeri romani minuscoli (pensateci: avete mai visto un'iscrizione romana in minuscolo? No, perché non esisteva il minuscolo). Peraltro il testo originale li richiede proprio maiuscoli. Per evitare qualunque problema nell'uso di `\MakeUppercase` (ne trovate un'ampia descrizione in Carlisle (2004)) usiamo il comando `\MakeTextUppercase` del pacchetto textcase.

Terminato il discorso sui capitoli, affrontiamo quello sui paragrafi. Qui il compito è più facile perché non esistono paragrafi non numerati e la struttura è semplicissima: numero di paragrafo imbandierato a sinistra, titolo del paragrafo allineato a quello del capitolo e numero di pagina imbandierato a destra, tutto in tondo.

```
1 \titlecontents{section}[0pc]
2   {}
3   {\makebox[6em][l]{
       \thecontentslabel}}
4   {}
5   {\hfill\thecontentspage}
```

Le uniche righe significative del codice sono dunque la 3 e la 5, rispettivamente per porre un box di larghezza fissa pari a quella del rispettivo box dei capitoli e contenente il numero di paragrafo, e per porre il numero di pagina tutto a destra.

4.3 Il sommario di *Microeconomia*

L'aspetto dell'indice di Varian (2012) è un po' più inconsueto rispetto ai canoni ai quali LaTeX ci ha abituato. Basta guardare la figura 8 per capire la differenza: i paragrafi e i sottoparagrafi non occupano una riga ciascuno ma sono tutti nello stesso capoverso, uno di seguito all'altro. Abbiamo già visto che titletoc espleta agevolmente anche questa funzione. Di nuovo, descriviamo le componenti dell'indice prima di vederne il codice.

Il titolo di un capitolo, in grassetto, occupa una riga a sé ed è preceduto dal numero del capitolo. Il numero di pagina è assente. In un blocco seguente al capitolo trovano posto i paragrafi e gli eventuali sotto(sotto)paragrafi. I paragrafi hanno il titolo in tondo e il numero di pagina a seguire in grassetto. Gli eventuali sottoparagrafi sono in corsivo e seguiti da un pallino (*bullet*) ma senza numero di pagina. Infine ci sono degli esempi riportati nell'indice, composti in corsivo senza numero di pagina e senza pallino. Li rappresentiamo al compilatore come `\subsubsection`. Per far comparire questa voce nell'indice dobbiamo modificare il numero di livello portando a 3 il valore immagazzinato nel contatore `tocdepth`.

Il codice usato è il seguente:

```
1  \titlecontents{chapter}[0pc]
2    {\vskip2\baselineskip
        \contentsmargin{0pt}\textbf
        {\textsf{\large
        \thecontentslabel}\qquad}}
3    {\bfseries\sffamily\large}
4    {}
5    {}
6  \titlecontents*{section}[0pc]
7    {}
8    {}
9    {}
10   {\quad\textbf{\thecontentspage
        }\hskip1.5em}
11 \titlecontents*{subsection}[0pc]
12   {}
13   {\itshape}
14   {\itshape}
15   {\quad\textbullet\quad}
16 \titlecontents*{subsubsection}[0
     pc]
17   {}
18   {\itshape}
19   {\itshape}
20   {\quad}
```

Il comando relativo al capitolo è molto semplice: come operazione preliminare lascia uno spazio verticale di `2\baselineskip` da quanto lo precede, quindi compone il numero di capitolo (riga 2) e poi il titolo (riga 3) in grassetto sans serif. Per i paragrafi e i sottoparagrafi bisogna cambiare la profondità come spiegato nel paragrafo 2. Il comando per comporre i paragrafi è asteriscato perché dobbiamo ottenere un blocco, non righe singole per titolo di paragrafo. Per comporre un paragrafo non dobbiamo far altro che aggiungere uno spazio e il numero di pagina in grassetto dopo il titolo (riga 10). I sottoparagrafi, che siano o no numerati, devono essere in corsivo (righe 13 e 14) e avere un pallino separato da adeguati spazi. La stessa cosa vale per i sottosottoparagrafi, da cui però eliminiamo il pallino e manteniamo lo spazio.

5 Conclusioni

Riprodurre lo stile di un editore è fondamentale per la corretta esecuzione di un lavoro su commissione. L'aspetto del sommario fa parte dello stile da riprodurre. Sebbene LaTeX metta a disposizione il comando `\addtocontents` per personalizzare entro un certo limite l'aspetto delle informazioni contenute nell'indice generale, può essere comodo avere un meccanismo più immediato e flessibile. Il pacchetto titletoc fornisce un siffatto meccanismo. Diventa quindi più facile personalizzare l'aspetto dei sommari, per non parlare della facilità di lettu-

Indice

1 Il mercato

Costruzione di un modello **1** Ottimizzazione ed equilibrio **2** La curva di domanda **3** La curva di offerta **5** Equilibrio di mercato **7** Statica comparata **8** Altri meccanismi per allocare appartamenti **11** *Il monopolista discriminante* • *Il monopolista puro* • *Controllo degli affitti* • Qual è il meccanismo migliore **13** Efficienza paretiana **14** Confronto tra i modi di allocare appartamenti **15** Equilibrio nel lungo periodo **16** Sommario **17** Domande **17**

2 Il vincolo di bilancio

Il vincolo di bilancio **19** Spesso due beni sono sufficienti **20** Proprietà dell'insieme di bilancio **20** Come varia la retta di bilancio **22** Il numerario **24** Tasse, sussidi e razionamento **25** *Esempio: il Food Stamp Program* Variazioni della retta di bilancio **29** Sommario **30** Domande **30**

3 Preferenze

Preferenze del consumatore **32** Assunzioni sulle preferenze **32** Curve di in-

FIGURA 8: Indice di VARIAN (2012) riprodotto con LaTeX.

ra del codice anche a distanza di tempo o da parte di una terza persona.

Nel presente articolo ho mostrato alcuni casi d'uso di libri pubblicati da editori reali e non tutti composti con TEX. Ho fornito alcuni esempi funzionanti di codice per riprodurre quanto più fedelmente possibile l'aspetto originale dei sommari. Questo mostra che l'uso di titletoc è assolutamente reale e non solo didattico, rendendolo un prodotto maturo per un eventuale uso di produzione e non solo di sperimentazione.

Riferimenti bibliografici

Bezos, J. (2011). *The titlesec, titleps and titletoc Packages*. URL `http://www.ctan.org`. Lo leggete dando `texdoc titletoc` al prompt del terminale o di DOS.

Carlisle, D. (2004). *The textcase package*. Lo leggete dando `texdoc textcase` al prompt del terminale o di DOS.

Kernighan, B. W. e Pike, R. (1999). *Programmazione nella pratica*. Addison-Wesley, Milano.

The LaTeX Project. «LaTeX Project Public License». Versione 1.3c.

Vacca, R. (1987). *Come imparare più cose e vivere meglio*. Oscar Mondadori. Arnoldo Mondadori Editore, Milano.

Varian, H. R. (2012). *Microeconomia*. Libreria Editrice Cafoscarina, Venezia.

▷ Gianluca Pignalberi
`g dot pignalberi at alice dot it`

Questa rivista è stata stampata
presso ERMES Servizi Editoriali Integrati S.R.L.
00040 Ariccia RM — via Quarto Negroni 15
su carta Arcoprint edizioni 1.3 85 g avorio
Copertina: Splendorgel 270 g avorio

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 Marzo 2014. 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@sssup.it`.

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 16, Ottobre 2013

5 Editoriale
Gianluca Pignalberi (?)

6 Scrivere la tesi di laurea in LATEX
Agostino De Marco

30 LATEX e le lingue romanze alpine: il piemontese
Claudio Beccari

39 LATEX e la documentazione di progetto
Roberto Giacomelli

52 Gli alfabeti romani di Francesco Griffo
Claudio Vincoletto

58 Il pacchetto minerva
Grazia Messineo, Salvatore Vassallo

68 Experiments with multitasking and multithreading in LUATEX
Luigi Scarso

84 Sommari (indici generali) personalizzati
Gianluca Pignalberi

9 771828 236001 40016