Numero 4
Ottobre
2007

ArsTEXnica

Rivista italiana di TEX e LATEX

Atti del GUIT *meeting 2007*

GUIT

http://www.guit.sssup.it/arstexnica

TEXnica Ars

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 un quota di iscrizione annuale di 25,00 € nel caso di persone fisiche o di 50,00 € nel caso di Enti o 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.guit.sssup.it`
`guit@sssup.it`

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

Codice ISSN 1828-2369

Stampata in Italia
Pisa: 13 Ottobre 2007

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 4, Ottobre 2007

Gruppo Utilizzatori Italiani di TEX

Editoriale

Massimiliano Dominici

Dopo due anni e quattro uscite, questo è l'ultimo numero che curo in qualità di direttore della rivista. Il mio impegno nel GUIT continua con l'incarico di presidente del Gruppo e con la volontà di contribuire ancora al lavoro della redazione, con altre mansioni. In queste occasioni, è d'uso stilare un piccolo bilancio del passato e trarre qualche auspicio per il futuro.

L'idea di realizzare una rivista che avesse il compito di raccogliere, dalla comunità TEX italiana, ma anche da quella internazionale, i contributi più significativi e di operare per diffonderli, è nata due anni fa, a conclusione del secondo convegno annuale del GUIT. Il fatto che in pochi mesi si sia riusciti a definire l'idea, organizzare la redazione e approntare il primo numero, va considerato, già di per sé, un risultato molto positivo. In questi due anni, poi, la rivista ha avuto modo di conquistarsi un posto alla pari con le altre iniziative del GUIT (penso al Forum e al convegno), affermandosi come fattore di crescita per il Gruppo. La redazione ha visto nuovi arrivi, oltre al gruppo dei collaboratori iniziali, segno sia della volontà di contribuire, da parte dei membri dell'associazione, sia della capacità di attrazione della rivista.

Naturalmente molte cose possono essere migliorate: l'organizzazione del lavoro all'interno della redazione, le modalità tecniche della produzione della rivista, la visibilità all'interno della comunità TEX internazionale. Sono certo che il nuovo direttore saprà affrontare al meglio queste problematiche e saprà rendere ArsTEXnica ancora più adatta a svolgere la sua missione statutaria.

Un ringraziamento caloroso va a tutte le persone che hanno collaborato, come membri della redazione, alla realizzazione di questi quattro numeri. Posso testimoniare che hanno sempre lavorato con impegno e dedizione, svolgendo i propri compiti in modo preciso e accurato.

Gli articoli di questo numero

Gli articoli contenuti in questo numero costituiscono gli Atti del Quarto Convegno Annuale del GUIT (GUIT*meeting2007*), che si svolge a Pisa, il 13 Ottobre 2007, presso l'Aula Magna della Scuola Superiore Sant'Anna. Anche quest'anno sono presenti interventi rappresentativi delle numerose aree d'impiego di TEX e destinati a diverse tipologie di utenti. Il Comitato Organizzatore ha cercato di raggruppare gli interventi in aree tematiche omogenee, nei limiti del possibile, e la rivista segue il medesimo ordine, nel presentare gli articoli.

Dopo l'introduzione, in cui il presidente uscente, Maurizio Himmelmann, e il nuovo presidente, Massimiliano Dominici, illustrano lo stato attuale del GUIT e i progetti per il futuro, la prima sessione mattutina offre un piccolo panorama di come TEX possa essere usato nella vita reale: dall'impiego individuale per la stesura del proprio curriculum vitæ (Lapo Mori e Maurizio Himmelmann, *Scrivere il curriculum vitæ con LATEX*), alla realizzazione di un prontuario di ingegneria (Claudio Beccari e Andrea Guadagni, *La progettazione di un'opera di consultazione*).

La seconda sessione mattutina è dedicata alla grafica. Massimo Caschili introduce all'uso di PSTricks, illustrandone la versatilità e offrendo una rapida carrellata delle estensioni che compongono questo sistema grafico. Luciano Battaia riporta la propria esperienza di impiego di LATEX e PSTricks nella scuola superiore, dicutendone le possibilità di diffusione. Infine Agostino De Marco mostra come è possibile realizzare illustrazioni tridimensionali da includere in documenti LATEX, utilizzando il programma Sketch, in grado di esportare codice interpretabile da PSTricks o PGF.

Nel pomeriggio, la prima sessione prevede l'intervento di Norbert Preining che spiega le ragioni e gli obiettivi alla base della nuova infrastruttura di TEXLive, infrastruttura che dovrebbe essere disponibile a partire dalla prossima versione. Segue Klaus Höppner che esporrà alcune strategie per ottenere tabelle tipograficamente gradevoli. Chiudono la prima sessione mattutina Gianluca Gorni e Stephane Matiz che illustrano TeXClipping, un pacchetto (realizzato dagli autori dell'intervento) per il software commerciale Mathematica, in grado di aggiungere ai grafici creati da questo programma etichette create con LATEX.

L'ultima sessione pomeridiana prevede due interventi centrati sui caratteri da stampa. Claudio Beccari presenta la sua versione, in formato METAFONT e TYPE1, dei font originariamente usati dal sistema SLITEX. Massimiliano Dominici illustra come sfruttare al massimo, con LATEX, un font TRUETYPE ricco di caratteristiche non standard, e offre come esempio i *Fell Types*.

Infine Jean-Michel Hufflen chiude anche questa sessione, e l'intero convegno, mostrando come è possibile scrivere riferimenti bibliografici adattabili a diversi schemi di citazioni.

Buona lettura.

▷ Massimiliano Dominici
mlgdominici@interfree.it

Programma del Convegno

Sessione Mattutina

09:00 – Registrazione

09:30 – Progetti e prospettive per GUIT

10:00 – Scrivere il curriculum vitæ con LaTeX
Lapo Mori, Università degli Studi di Pisa
Maurizio Himmelmann, Scuola Superiore Sant'Anna

10:30 – La progettazione di un'opera di consultazione
Claudio Beccari, Politecnico di Torino
Andrea Guadagni, Hoepli Editore, Milano

11:00 – Coffee break

11:30 – Introduzione a PSTricks
Massimo Caschili

12:00 – LaTeX nella scuola media superiore: applicazioni didattiche con PSTricks
Luciano Battaia, Liceo Scientifico Grifoletti, Pordenone

12:30 – Illustrazioni tridimensionali con Sketch/LaTeX/PSTricks/Ti*kz* nella dinamica del volo
Agostino De Marco, Università degli Studi di Napoli "Fedrico II"

13:00 – Pausa Pranzo

Sessione Pomeridiana

15:00 – TeXLive's infrastructure
Norbert Preining, Vienna University of Technology

15:30 – Typesetting tables with LaTeX
Klaus Höppner, DANTE e.V.

16:00 – Inserire equazioni LaTeX in grafici di Mathematica
Gianluca Gorni, Universitá degli Studi di Udine
Stephane Matiz, Universitá degli Studi di Udine

16:30 – Coffee break

17:00 – I font per le slide LaTeX risuscitati
Claudio Beccari, Politecnico di Torino

17:30 – Utilizzo di caratteri TrueType con LaTeX. Un esempio pratico i Fell Types
Massimiliano Dominici

18:00 – Guidelines for bibliographocal citations in LaTeX
Jean Michel Hufflen, Université de Franche Comté

18:30 – LaTeX Help Desk (esperti di LaTeX a vostra disposizione)

19:00 – Chiusura dei lavori e cena sociale

Scrivere il curriculum vitæ con LaTeX

*Lapo F. Mori, Maurizio W. Himmelmann**

Sommario

Quest'articolo si propone di illustrare alcune soluzioni atte a redigere un curriculum vitæ con LaTeX, passando per una rassegna critica di pacchetti e classi.

Abstract

This paper presents the tools that are currently available to prepare a curriculum vitæ with LaTeX with a critical analysis of packages and classes.

1 Premessa

Il presente articolo tratta la composizione del curriculum vitæ sia in termini generali che, nello specifico, con LaTeX. La prima parte del documento (par. 2 e 3) ne esamina le linee guida di composizione, che possono essere sfruttate con qualunque editor di testo, mentre la seconda parte (par. 4 e 6) richiede conoscenze elementari di LaTeX, che possono essere acquisite consultando una guida di base qualunque (Baudoin, 1998; Beccari, 2007; Oetiker *et al.*, 2000; The Tutorial Team, 2000) o un manuale cartaceo (Goossens *et al.*, 1995; Kopka e Daly, 1995).

L'articolo, dopo aver analizzato i vari stili e le strutture di composizione (par. 2), fornirà tutte le indicazioni necessarie a scrivere un buon curriculum vitæ (par. 3). Tali indicazioni giustificheranno, poi, l'uso di LaTeX come sistema di composizione per il proprio curriculum vitæ e, a tal proposito, si tratteranno le classi, i pacchetti disponibili (par. 4) e i modi per personalizzarne il codice (par. 6).

2 Introduzione

Il curriculum vitæ è un documento che contiene l'elenco delle esperienze lavorative e accademiche di un individuo e di norma viene redatto per presentare se stessi così da ottenere un'intervista con potenziali datori di lavoro. Esso, infatti, rappresenta la prima fonte informativa in base alla quale il datore di lavoro (o il selezionatore del personale) decide se convocare o meno il candidato, per valutarne l'assunzione.

Vi è un'abbondante letteratura, sia in italiano che in inglese, che illustra le migliori strategie per scrivere un curriculum vitæ (Adani, 2001, 2002; Amadori e De Giuli, 1993; Annovazzi, 1998; Anthony e Roe, 1998; Barbasio e Piccardo, 2003; Biglino, 1993; Boschi e Lepore, 2001; Bruni *et al.*, 1997; Buratto, 2007; Cama, 1996; Catalani, 1998; Consul e Di Fresco, 2003; Crimini e Giusti, 2002; Demi e Santonocito, 1997, 2004; Fiorini, 1995; Jackson e Geckeis, 2003; Meyer, 1991; Passerini, 1999; Rustico, 2000; Vascon, 2000) e il presente articolo non pretende di essere completo ed esaustivo alla pari di quelli appena citati, ma si limita a fornire linee guida di massima.

2.1 Nome

I termini usati per il curriculum vitæ sono diversificati in base alla lingua dell'autore. In italiano, la parola latina *curriculum* (derivata da *currere* che significa "correre") è entrata attraverso la locuzione *curriculum vitæ* sul finire del 1800 con il proprio significato letterale di "córso della vita". La maggior parte dei dizionari italiani considerano la locuzione latina *curriculum vitæ* un sostantivo maschile invariante (Devoto e Oli, 2008; Duro, 1986; Patota, 2006; Zingarelli, 1984) in accordo con le regole che le parole in lingua straniera non si alterano al plurale e che le parole terminanti in consonante non mutano al plurale. Tuttavia altri, tra cui Setti (2002), ottengono il plurale di *curriculum* con la tipica terminazione in -a del plurale del neutro latino (vedi al riguardo l'appendice A).

Nonostante esista anche la forma *adattata* all'italiano "curricolo" con il suo plurale regolare "curricoli", il termine originario latino continua ad essere quello più utilizzato.

Il curriculum vitæ viene così chiamato (talvolta semplicemente "curriculum" oppure abbreviato come "CV") in Europa, Nuova Zelanda, Canada francese e in altri stati del Commonwealth britannico. In nord America (Stati Uniti e Canada inglese) e nelle Filippine si distingue tra *résumé*,[1] quando il documento è destinato al mondo del lavoro privato, e curriculum vitæ (talvolta chiamato semplicemente *vita*) nel mondo accademico. In Australia e in India curriculum vitæ e *résumé* vengono usati indistintamente.

*Desideriamo ringraziare Claudio Beccari ed in particolare Valeria Angeli e Caterina Mori per gli utili suggerimenti forniti nelle fasi di stesura e revisione di questo articolo. Si ringrazia anche lo staff di ArsTEXnica per l'ottimo lavoro di recensione svolto, che ha contribuito a migliorare notevolmente la qualità del presente articolo.

1. La parola *résumé*, entrata in uso in diverse lingue anglofone, è il participio passato del verbo *résumer* che in francese significa riassumere e a sua volta deriva dal latino *resumere*. Si trova comunemente anche scritta come *resumé* e *resume*.

2.2 Contenuto

2.2.1 Curriculum vitæ

Il curriculum vitæ contiene l'elenco completo delle esperienze professionali e accademiche, l'elenco delle pubblicazioni, dei contributi apportati e dei risultati ottenuti. Oltre a queste informazioni, i candidati possono aggiungere informazioni personali, non legate direttamente alla loro vita professionale, in modo da fornire al datore di lavoro un'immagine più completa di se stessi.

2.2.2 Résumé

Il *résumé* è generalmente più breve del curriculum vitæ e si limita ad una o due pagine in cui vengono evidenziate solo le esperienze e le credenziali che l'autore ritiene più rilevanti per la domanda di lavoro che sta presentando. Di solito un *résumé* contiene parole chiave precise atte a catturare l'attenzione dei potenziali datori di lavoro.

2.3 Stile

2.3.1 Cronologico

Lo stile cronologico prevede che le esperienze lavorative e accademiche del candidato vengano elencate in ordine cronologico inverso.[2] Questo stile è il più diffuso e serve ad evidenziare l'evoluzione professionale del candidato, ripercorrendo le tappe in ordine cronologico.

2.3.2 Funzionale

Lo stile funzionale prevede che le esperienze di lavoro siano raggruppate per aree tematiche ed è usato per evidenziare le competenze e l'esperienza maturata dal candidato in alcune aree specifiche.

2.3.3 Misto

Esiste anche uno stile misto, in cui le aree tematiche (funzionale) sono esposte in ordine cronologico inverso (cronologico).

2.4 Struttura

La struttura ed i contenuti di un curriculum vitæ possono variare enormemente in funzione del contesto. In generale è possibile distinguere tra informazioni *essenziali* e *facoltative*.

2.4.1 Informazioni essenziali

- **Dati anagrafici**: nome, sesso, data e luogo di nascita, cittadinanza, stato civile, indirizzo, numeri di telefono e fax, email, sito internet.[3]
- **Istruzione**: include una lista in ordine cronologico inverso delle qualifiche e dei titoli ottenuti, sia accademici che professionali, compresa l'appartenenza ad associazioni di categoria. È possibile aggiungere dettagli sugli esami sostenuti, se rilevanti per la particolare domanda di lavoro presentata.
- **Esperienze lavorative**: salvo nel caso in cui il candidato stia per affrontare il primo impiego, tutte le esperienze lavorative (compresa quella attuale) vengono elencate in ordine cronologico inverso. Per ogni tappa della propria carriera dovrebbero essere indicati i risultati raggiunti, piuttosto che le mansioni svolte. Per le esperienze lavorative più importanti dovrebbero essere presenti: il concetto, la pianificazione, i risultati e i ruoli ricoperti.[4]

2.4.2 Informazioni facoltative

- **Profilo personale**: sebbene non sia frequente, nel curriculum vitæ può essere inserito un paragrafo che descrive brevemente il candidato. Scritto in prima o terza persona, serve a delineare le migliori qualità e capacità del candidato. La descrizione dovrebbe essere basata su meri fatti ed escludere opinioni sulle qualità del candidato. Se presente, di solito è all'apertura del curriculum vitæ.
- **Obiettivi personali**: talvolta può essere presente una frase che descrive l'obiettivo professionale del candidato. Sebbene sia poco diffuso, viene talvolta richiesto per fare una selezione veloce dei candidati, nei casi in cui il loro numero sia molto elevato.
- **Fotografia**: la presenza di una fotografia dipende molto dal contesto geografico. In Germania la fotografia è richiesta nel curriculum vitæ da molto tempo. In India è uso comune inserire una fotografia qualora si faccia domanda per un lavoro in cui è richiesto contatto con il pubblico. Negli Stati Uniti è fortemente sconsigliato aggiungere una foto perché potrebbe suggerire che il datore di lavoro sia interessato all'aspetto fisico del candidato (età, razza, sesso, ecc.). In Italia non ci sono consuetudini al riguardo e quindi la scelta è lasciata al candidato.
- **Conoscenza di lingue straniere**: è opportuno indicare quali lingue sono conosciute dal candidato e può essere specificata la capacità di comunicazione scritta e orale. Se presenti, è consigliabile riportare i risultati degli esami di lingua straniera sostenuti.

2. Più raramente l'ordine cronologico può essere diretto.

3. Non tutti i dati qui elencati sono generalmente necessari. In certi casi è esplicitamente richiesto che alcuni di questi vengano omessi (ad esempio il sesso e la data di nascita), così da evitare possibili sospetti di discriminazione; negli Stati Uniti, ad esempio, c'è particolare sensibilità verso queste tematiche, pertanto non vi è la pretesa che le informazioni anagrafiche compaiano nel curriculum vitæ.

4. In passato era consigliato anche specificare se era stato assolto o meno il servizio militare, ma in Italia, vista l'avvenuta sospensione del servizio di leva, questa informazione è caduta in disuso.

- **Conoscenze informatiche**: l'elenco delle conoscenze informatiche era molto importante negli anni '80 ma è stato considerato superfluo (in quanto ormai dato per scontato) già dagli anni '90. Può essere, tuttavia, importante nel caso di competenze a riguardo di software tecnici, linguaggi di programmazione o comunque per quanto concerne strumenti che non rientrino nell'ambito delle conoscenze informatiche di base.
- **Attività extraprofessionali**: interessi, hobby, sport, etc.

2.4.3 Formato europeo

La Commissione Europea ha proposto un modello di curriculum vitæ standardizzato che è stato adottato dal Parlamento Europeo e dal Consiglio il 15 dicembre 2004. Tale modello, chiamato Europass CV ha sostituito il CV Europeo che era stato introdotto nel 2002 e fornisce linee guida sia sullo stile che sui contenuti del curriculum vitæ. Il modello prevede le seguenti sezioni obbligatorie (Comunità Europea, 2003):

- informazioni personali: nome, indirizzo, telefono, fax, email, nazionalità, data di nascita e sesso;
- impiego ricercato/settore di competenza;
- esperienza professionale: date, funzione o posto occupato, principali mansioni e responsabilità, nome e indirizzo del datore di lavoro, tipo o settore d'attività;
- capacità e competenze personali: madrelingua, altre lingue; il testo fornisce una griglia per l'autovalutazione della capacità di capire, parlare e scrivere nelle lingue straniere conosciute;
- istruzione e formazione: date, certificato o diploma ottenuto, principali materie/competenze professionali apprese, nome e tipo d'istituto di istruzione o formazione;
- descrizione dettagliata delle capacità e competenze acquisite nel percorso formativo, della carriera professionale o della vita quotidiana;

e le seguenti sezioni facoltative

- capacità e competenze sociali;
- capacità e competenze organizzative;
- capacità e competenze tecniche;
- capacità e competenze informatiche;
- capacità e competenze artistiche;
- altre capacità e competenze;
- patente;
- informazioni complementari;
- allegati.

2.4.4 Lettera di presentazione e motivazione

La lettera di presentazione (*cover letter*) viene generalmente inviata all'azienda assieme al curriculum vitæ. Lo scòpo è quello di presentare la candidatura rimandando poi al curriculum vitæ per maggiori dettagli sulle esperienze formative e lavorative. La lettera deve contenere una brevissima presentazione di se stessi e le motivazioni che spingono a candidarsi per quell'impiego o ruolo. In questi casi, una buona strategia consiste nel dimostrare di avere informazioni sull'azienda e sull'eventuale ruolo da ricoprire, così come mettere in luce le proprie competenze ed esperienze. Le esperienze rilevanti devono essere evidenziate, facendo poi riferimento a specifiche parti del curriculum vitæ per maggiori dettagli. È consigliabile non tralasciare le competenze acquisite o i ruoli di responsabilità e i successi ottenuti in attività che possono essere importanti per l'azienda. Si raccomanda, infine, di firmare sempre in calce la lettera.

3 Scrivere un buon curriculum vitæ

3.1 Importanza del curriculum vitæ

Il curriculum vitæ costituisce spesso il primo contatto con il futuro datore di lavoro e per questo motivo la sua redazione risulta essere una tappa importante nella ricerca di un impiego. Il curriculum vitæ deve essere curato sia nella *forma* che nei *contenuti* e deve essere privo sia di errori ortografici che di punteggiatura. Prima di consegnarlo all'azienda, è consigliabile proporne la lettura a terze persone, così da assicurarsi che il contenuto sia davvero chiaro. Il contenuto deve essere adeguato al profilo richiesto. Spesso è conveniente concentrare l'attenzione sulle esperienze che danno valore aggiunto alla candidatura, ottenendo un curriculum vitæ più corto: in questo modo si evita che nello sfogliare il documento non vengano, poi, tralasciate le parti più importanti ai fini della candidatura. Per adattare meglio il curriculum vitæ alla candidatura è consigliabile raccogliere informazioni sull'azienda alla quale si presenterà poi la domanda.

3.2 Essere sintetici

Un male comune di molti curriculum vitæ è la *lunghezza inutile*. Molti candidati hanno la pessima abitudine di includere nel proprio tutto quello che hanno fatto nella vita, compresi particolari del tutto insignificanti dal punto di vista del datore di lavoro, invece di selezionare solo gli aspetti rilevanti. In generale è sempre bene ridurre al minimo la lunghezza del curriculum vitæ in modo

che gli aspetti più significativi siano messi in risalto piuttosto che diluiti in un *mare magnum* di esperienze scorrelate dall'impiego per il quale ci si sta candidando.

Il consiglio migliore riguardo la lunghezza del curriculum vitæ è di informarsi preventivamente su quale sia la lunghezza che l'azienda si aspetta dai candidati. Nella maggior parte dei casi non si devono superare le due pagine (talvolta solo una) ma in altri casi il curriculum vitæ deve essere dettagliato.

Un'alternativa, seguita dallo stesso Donald Knuth,[5] consiste nel preparare due versioni del curriculum vitæ, una sintetica ed una estesa. La versione sintetica può contenere un link a quella estesa in modo che il datore di lavoro possa esaminarle entrambe se lo ritiene necessario.

3.3 Attirare l'attenzione

Il curriculum vitæ svolge gran parte del suo ruolo nelle fasi iniziali del processo di assunzione, in cui il datore di lavoro esamina molte domande dedicando ad ognuna poco tempo. Per questo motivo è importante che il documento attiri l'attenzione ed emerga tra quelli dei concorrenti. Esistono svariati modi per differenziarsi in positivo: con un elegante stile tipografico, selezionando il contenuto in modo adeguato, ecc... Saper presentare le proprie conoscenze e capacità in modo chiaro ed ordinato costituisce, già di per sé, un vantaggio iniziale nei confronti dei concorrenti.

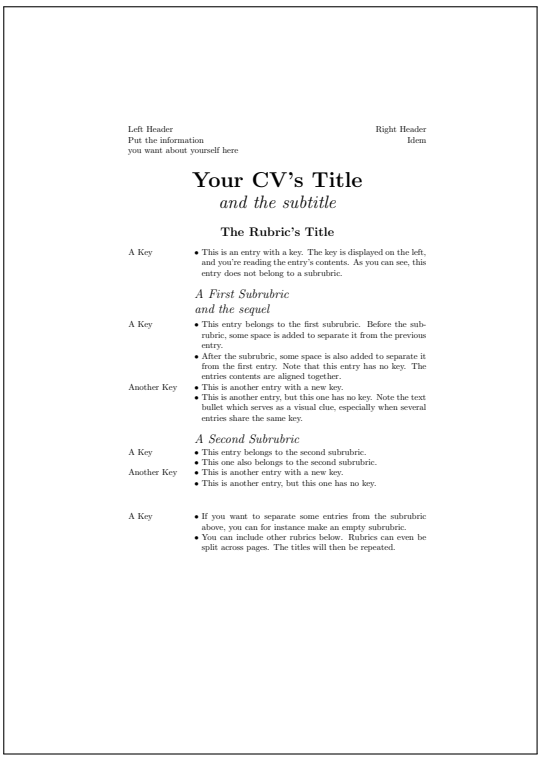
3.4 LaTeX

LaTeX, grazie all'elevata qualità dei documenti prodotti, si presta particolarmente bene per la redazione di un curriculum vitæ che sarà il proprio biglietto di visita per il futuro datore di lavoro. Anche la semplicità di redazione e la possibilità di utilizzare automatismi e stili predefiniti giocano un ruolo determinante in favore di questa scelta. I paragrafi seguenti (par. 4 e 6) forniscono le basi per scrivere il curriculum vitæ con questo programma.

4 Classi e pacchetti LaTeX

4.1 CurVe

CurVe, scritta da Didier Verna nel 2000, rientra di diritto tra le classi più complete per la realizzazione di curriculum vitæ e sicuramente è quella che permette il maggiore grado di flessibilità. Sua caratteristica peculiare è la possibilità di ripartire il proprio curriculum vitæ in diverse sezioni, dette rubriche. Spesso infatti ci si accorge che il curriculum vitæ, già preparato con tanta cura, non si adatta a particolari situazioni che necessitano di enfasi su determinati argomenti piuttosto che su altri. CurVe permette, quindi, di creare un curriculum vitæ con una struttura modulare, consentendo all'utente di adattarlo rapidamente e con pochissimo sforzo alle caratteristiche che desidera trasmettere all'azienda. Ogni rubrica racchiude un argomento principale, come ad esempio "Educazione" o "Esperienze Lavorative" (o anche solo il medesimo argomento, ma descritto in modi differenti). La combinazione di queste rubriche permette di comporre il curriculum vitæ così da adattarlo al profilo richiesto dall'azienda, senza dover cambiare il contenuto del file `.tex`. Le singole rubriche (o sottorubriche) possono essere selettivamente richiamate nel file principale con il comando `\makerubric{rubric}`.


Left Header
Put the information you want about yourself here
Right Header
Idem

Your CV's Title
and the subtitle

The Rubric's Title

A Key
• This is an entry with a key. The key is displayed on the left, and you're reading the entry's contents. As you can see, this entry does not belong to a subrubric.

A First Subrubric
and the sequel

A Key
• This entry belongs to the first subrubric. Before the subrubric, some space is added to separate it from the previous entry.
• After the subrubric, some space is also added to separate it from the first entry. Note that this entry has no key. The entries contents are aligned together.

Another Key
• This is another entry with a new key.
• This is another entry, but this one has no key. Note the text bullet which serves as a visual clue, especially when several entries share the same key.

A Second Subrubric

A Key
• This entry belongs to the second subrubric.
• This one also belongs to the second subrubric.

Another Key
• This is another entry with a new key.
• This is another entry, but this one has no key.

A Key
• If you want to separate some entries from the subrubric above, you can for instance make an empty subrubric.
• You can include other rubrics below. Rubrics can even be split across pages. The titles will then be repeated.

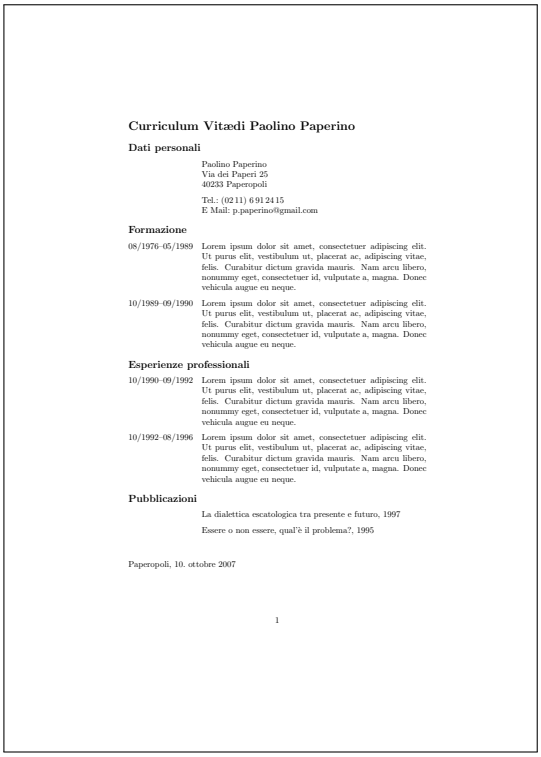
FIGURA 1: Classe CurVe.

La sintassi utilizzata è estremamente semplice e non richiede, (almeno nella fase iniziale) di dovere consultare continuamente la documentazione, che è comunque abbastanza completa e ricca di informazioni. Una lettura più approfondita della documentazione è, invece, indispensabile qualora ci si inoltri nei tortuosi (ma gratificanti) meandri della personalizzazione.

Per scelta dell'autore del pacchetto non viene incoraggiato l'uso di file bibliografici esterni al curriculum vitæ stesso, ma ne resta comunque possibile l'utilizzo.

Analogamente a quanto permesso da altre classi di questo genere, è prevista la possibilità di inserire la propria fotografia. Eccellente il supporto multilingua, così come la disponibilità di *template* da personalizzare a proprio piacere. All'interno della documentazione sono forniti anche degli *script* per compilare automaticamente le differenti tipologie di curriculum vitæ.

5. `http://www-cs-faculty.stanford.edu/~knuth/vita.html`



Curriculum Vitædi Paolino Paperino

Dati personali

Paolino Paperino
Via dei Paperi 25
40233 Paperopoli

Tel.: (02 11) 6 91 24 15
E Mail: p.paperino@gmail.com

Formazione

08/1976–05/1989 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque.

10/1989–09/1990 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque.

Esperienze professionali

10/1990–09/1992 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque.

10/1992–08/1996 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque.

Pubblicazioni

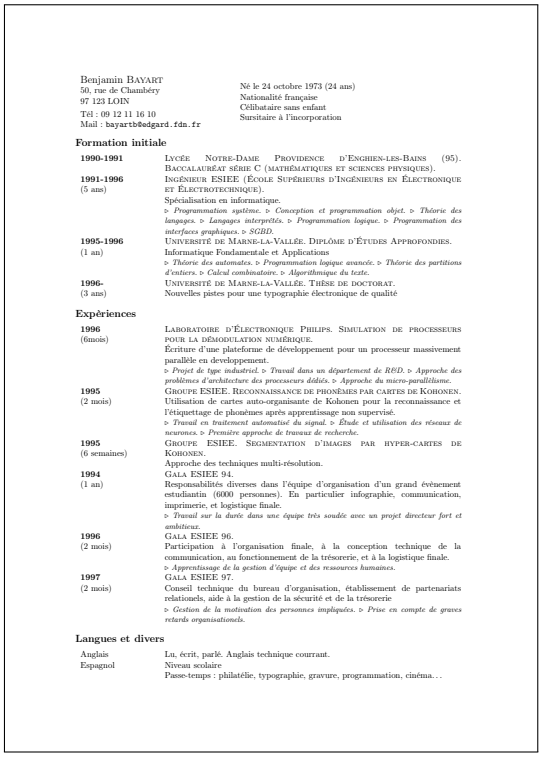
La dialettica escatologica tra presente e futuro, 1997

Essere o non essere, qual'è il problema?, 1995

Paperopoli, 10. ottobre 2007

1

FIGURA 2: Pacchetto currvita.



Benjamin BAYART
50, rue de Chambéry
97 123 LOIN
Tél : 09 12 11 16 10
Mail : bayartb@edgard.fdn.fr

Né le 24 octobre 1973 (24 ans)
Nationalité française
Célibataire sans enfant
Sursitaire à l'incorporation

Formation initiale

1990-1991 LYCÉE NOTRE-DAME PROVIDENCE D'ENGHIEN-LES-BAINS (95). BACCALAURÉAT SÉRIE C (MATHÉMATIQUES ET SCIENCES PHYSIQUES).

1991-1996 (5 ans) INGÉNIEUR ESIEE (ÉCOLE SUPÉRIEURS D'INGÉNIEURS EN ÉLECTRONIQUE ET ÉLECTROTECHNIQUE).
Spécialisation en informatique.
▷ *Programmation système.* ▷ *Conception et programmation objet.* ▷ *Théorie des langages.* ▷ *Langages interprétés.* ▷ *Programmation logique.* ▷ *Programmation des interfaces graphiques.* ▷ *SGBD.*

1995-1996 (1 an) UNIVERSITÉ DE MARNE-LA-VALLÉE. DIPLÔME D'ÉTUDES APPROFONDIES.
Informatique Fondamentale et Applications
▷ *Théorie des automates.* ▷ *Programmation logique avancée.* ▷ *Théorie des partitions d'entiers.* ▷ *Calcul combinatoire.* ▷ *Algorithmique du texte.*

1996- (3 ans) UNIVERSITÉ DE MARNE-LA-VALLÉE. THÈSE DE DOCTORAT.
Nouvelles pistes pour une typographie électronique de qualité

Expèriences

1996 (6mois) LABORATOIRE D'ÉLECTRONIQUE PHILIPS. SIMULATION DE PROCESSEURS POUR LA DÉMODULATION NUMÉRIQUE.
Écriture d'une plateforme de développement pour un processeur massivement parallèle en developpement.
▷ *Projet de type industriel.* ▷ *Travail dans un département de R&D.* ▷ *Approche des problèmes d'architecture des processeurs dédiés.* ▷ *Approche du micro-parallélisme.*

1995 (2 mois) GROUPE ESIEE. RECONNAISSANCE DE PHONÈMES PAR CARTES DE KOHONEN.
Utilisation de cartes auto-organisante de Kohonen pour la reconnaissance et l'étiquetage de phonèmes après apprentissage non supervisé.
▷ *Travail en traitement automatisé du signal.* ▷ *Étude et utilisation des réseaux de neurones.* ▷ *Première approche de travaux de recherche.*

1995 (6 semaines) GROUPE ESIEE. SEGMENTATION D'IMAGES PAR HYPER-CARTES DE KOHONEN.
Approche des techniques multi-résolution.

1994 (1 an) GALA ESIEE 94.
Responsabilités diverses dans l'équipe d'organisation d'un grand évènement estudiantin (6000 personnes). En particulier infographie, communication, imprimerie, et logistique finale.
▷ *Travail sur la durée dans une équipe très soudée avec un projet directeur fort et ambitieux.*

1996 (2 mois) GALA ESIEE 96.
Participation à l'organisation finale, à la conception technique de la communication, au fonctionnement de la trésorerie, et à la logistique finale.
▷ *Apprentissage de la gestion d'équipe et des ressources humaines.*

1997 (2 mois) GALA ESIEE 97.
Conseil technique du bureau d'organisation, établissement de partenariats relationels, aide à la gestion de la sécurité et de la trésorerie
▷ *Gestion de la motivation des personnes impliquées.* ▷ *Prise en compte de graves retards organisationels.*

Langues et divers

Anglais — Lu, écrit, parlé. Anglais technique courrant.
Espagnol — Niveau scolaire
Passe-temps : philatélie, typographie, gravure, programmation, cinéma...

FIGURA 3: Pacchetto ESIEEcv.

Difetti

Essendo implementato sul pacchetto longtable, Curve risente dei limiti di quest'ultimo: l'inserimento delle diverse rubriche, essendo concepite come blocchi individuali ed indipendenti, richiede sempre un minimo di adattamento da parte dell'utente. Infine, il layout complessivo, per quanto pulito e razionale, non rientra tra le soluzioni più accattivanti per la redazione di un curriculum vitæ.

4.2 currvita

currvita è un pacchetto scritto da Axel Reichert nell'ormai lontano 1999 ed era presentato come alternativa agli unici pacchetti allora disponibili (vita, resumee e tabularx). Proposto anche all'interno della *suite* di classicthesis, permette di redigere un curriculum vitæ con una struttura molto semplice ed essenziale, senza possibilità di effettuare personalizzazioni che vadano oltre il semplice posizionamento della data e lo spessore dell'interlinea tra le sezioni. Nel caso in cui l'utente desiderasse modificare il layout standard, avrebbe ampia possibilità di scelta entro un lungo elenco di `\renewcommand`.

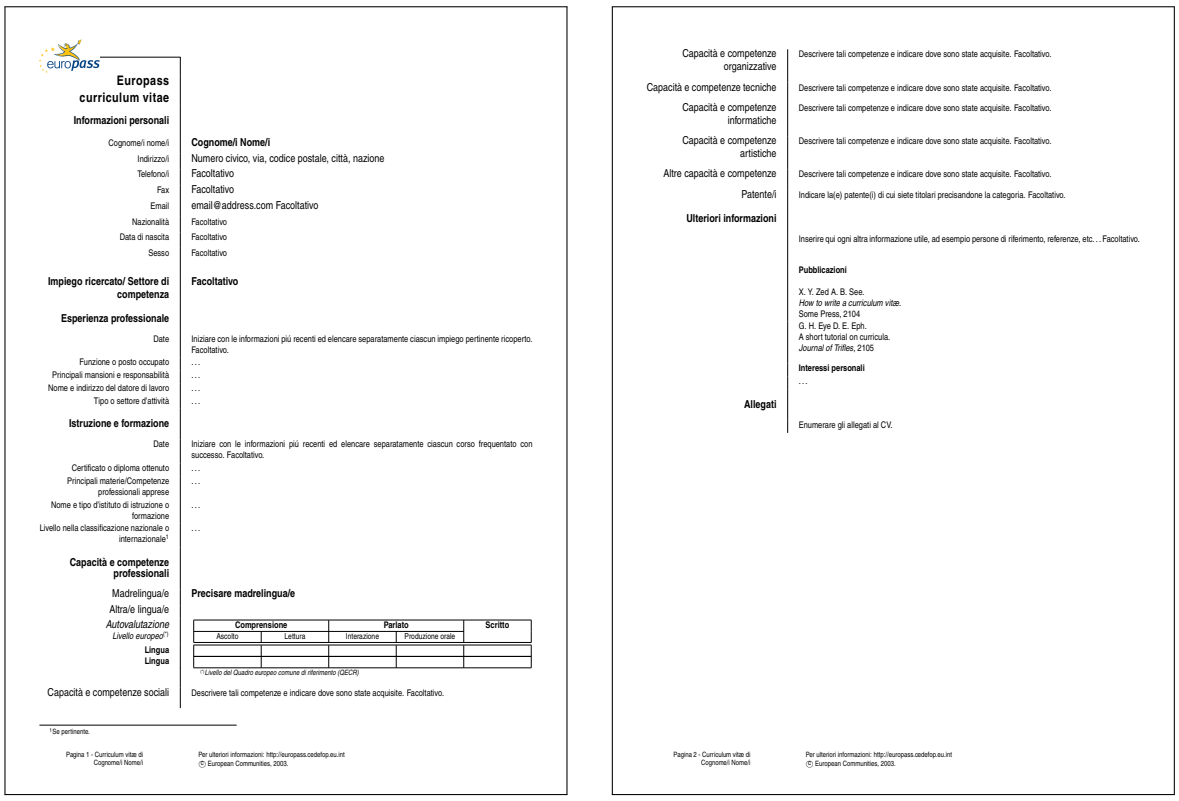
La documentazione è abbastanza completa ed include, oltre ad esaurienti spiegazioni sulla sintassi, anche le sempre utili norme tipografiche su come scrivere il curriculum. Il pacchetto resta un buon metro di paragone per apprezzare i progressi compiuti finora da altre classi e la ricchezza di proposte attualmente disponibili.

4.3 ESIEEcv

Realizzato nel 1997 da Benjamin Bayart per uso personale, e successivamente rilasciato su CTAN, ESIEEcv è un pacchetto scritto interamente in francese (inclusa documentazione e comandi dedicati) e non supporta nativamente altri linguaggi. Nel caso lo si volesse usare con un'altra lingua occorre direttamente modificare il codice sorgente della classe. Il codice del *template* è abbastanza intuitivo, prevedendo l'uso di numerosi ambienti dedicati (rubrique e sousrubrique) all'interno dei quali strutturare i contenuti con appositi comandi (`\Titre`, `\Lieu`, ecc.). Il layout è sufficientemente chiaro.

4.4 europecv

L'undici marzo 2002 la Commissione Europea, nel tentativo di uniformare le informazioni curriculari dei cittadini comunitari, ha creato e messo a disposizione sul web un *template* per redigere il proprio curriculum vitæ. Come spesso purtroppo accade, nel mettere in pratica questa lodevole iniziativa ci si è, però, completamente dimenticati dell'esistenza di altri sistemi di typesetting non proprietari, rilasciando il modello ufficiale nel solo formato `.doc`. Nicola Vitacolonna ha implementato la classe LaTeX europecv in modo da essere quanto più simile possibile al modello della Commissione Europea, senza sacrificarne però la flessibilità. Solo un occhio più che attento sarà in grado di distinguere le differenze col modello originale (anche se quest'improbabile ipotesi si dovesse verificare, il modello originale uscirebbe comunque perdente dall'impari confronto per qualità tipografica con



europass

Europass curriculum vitae

Informazioni personali

Cognome/i nome/i	**Cognome/i Nome/i**
Indirizzo/i	Numero civico, via, codice postale, città, nazione
Telefono/i	Facoltativo
Fax	Facoltativo
Email	email@address.com Facoltativo
Nazionalità	Facoltativo
Data di nascita	Facoltativo
Sesso	Facoltativo

Impiego ricercato/ Settore di competenza **Facoltativo**

Esperienza professionale

Date	Iniziare con le informazioni più recenti ed elencare separatamente ciascun impiego pertinente ricoperto. Facoltativo.
Funzione o posto occupato	…
Principali mansioni e responsabilità	…
Nome e indirizzo del datore di lavoro	…
Tipo o settore d'attività	…

Istruzione e formazione

Date	Iniziare con le informazioni più recenti ed elencare separatamente ciascun corso frequentato con successo. Facoltativo.
Certificato o diploma ottenuto	…
Principali materie/Competenze professionali apprese	…
Nome e tipo d'istituto di istruzione o formazione	…
Livello nella classificazione nazionale o internazionale[1]	…

Capacità e competenze professionali

Madrelingua/e **Precisare madrelingua/e**

Altra/e lingua/e

Autovalutazione

	Comprensione		Parlato		Scritto
Livello europeo[1]	Ascolto	Lettura	Interazione	Produzione orale	
Lingua					
Lingua					

[1] *Livello del Quadro europeo comune di riferimento (QECR)*

Capacità e competenze sociali	Descrivere tali competenze e indicare dove sono state acquisite. Facoltativo.
Capacità e competenze organizzative	Descrivere tali competenze e indicare dove sono state acquisite. Facoltativo.
Capacità e competenze tecniche	Descrivere tali competenze e indicare dove sono state acquisite. Facoltativo.
Capacità e competenze informatiche	Descrivere tali competenze e indicare dove sono state acquisite. Facoltativo.
Capacità e competenze artistiche	Descrivere tali competenze e indicare dove sono state acquisite. Facoltativo.
Altre capacità e competenze	Descrivere tali competenze e indicare dove sono state acquisite. Facoltativo.
Patente/i	Indicare la(e) patente(i) di cui siete titolari precisandone la categoria. Facoltativo.

Ulteriori informazioni

Inserire qui ogni altra informazione utile, ad esempio persone di riferimento, referenze, etc… Facoltativo.

Pubblicazioni

X. Y. Zed A. B. See.
How to write a curriculum vitæ.
Some Press, 2104
G. H. Eye D. E. Eph.
A short tutorial on curricula.
Journal of Trifles, 2105

Interessi personali

…

Allegati

Enumerare gli allegati al CV.

[1] Se pertinente.

FIGURA 4: Classe europecv.

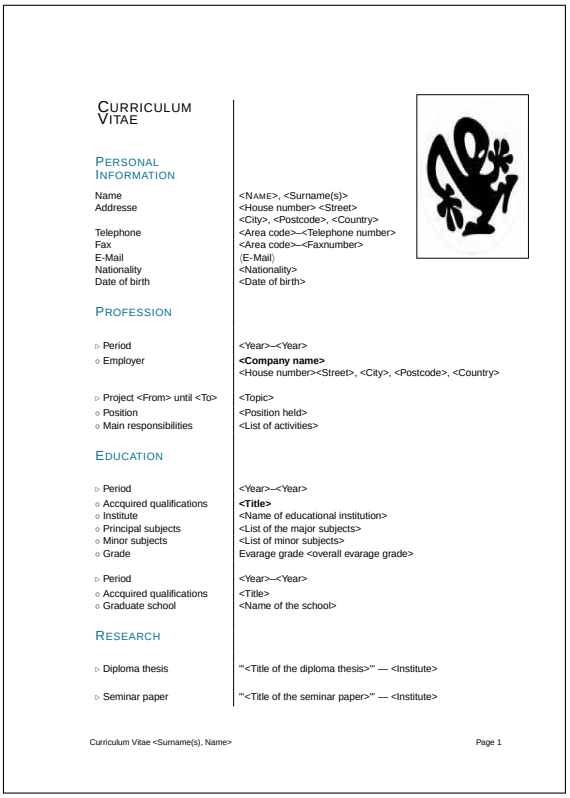
il presente). Il layout del curriculum vitæ è chiaro e gradevole ed ha il vantaggio di prevedere anche uno schema in cui poter indicare il grado di conoscenza delle lingue secondo il *Common European Framework of Reference* (CEFR).

La classe è ormai consolidata ed ha superato i piccoli difetti delle prime versioni. Essa mette a disposizione dell'utente una ricca serie di comandi per comporre il curriculum vitæ. È possibile inserire la propria fotografia così come effettuare un minimo di personalizzazione del layout (senza uscire dal modello della CE). La documentazione è molto completa e ben realizzata, infatti fornisce in modo chiaro e immediato tutte le informazioni sulle funzionalità di ciascun comando. Al termine è inserita anche un'utile descrizione dei vari livelli di conoscenza delle lingue straniere, così come definito dal CEFR.

Il supporto multilingua è molto valido, così come la possibilità di optare per un logo a colori o in tonalità di grigio. È stata anche prevista la possibilità di collegare un file `.bib` per inserire l'elenco delle pubblicazioni. Vengono inoltre forniti sia i file sorgenti che i compilati di numerosi *template*.

4.5 ecv

ecv, pacchetto scritto da Christoph Neumann e Bernd Haberstrumpf nel 2007, fornisce un altro ambiente basato su tabular per scrivere il curriculum vitæ seguendo le indicazioni fornite dalla Commissione Europea. Il risultato è decisamente inferiore per qualità rispetto alla classe europecv,



CURRICULUM VITAE

PERSONAL INFORMATION

Name	<NAME>, <Surname(s)>
Addresse	<House number> <Street> <City>, <Postcode>, <Country>
Telephone	<Area code>–<Telephone number>
Fax	<Area code>–<Faxnumber>
E-Mail	(E-Mail)
Nationality	<Nationality>
Date of birth	<Date of birth>

PROFESSION

▹ Period	<Year>–<Year>
◦ Employer	**<Company name>** <House number><Street>, <City>, <Postcode>, <Country>
▹ Project <From> until <To>	<Topic>
◦ Position	<Position held>
◦ Main responsibilities	<List of activities>

EDUCATION

▹ Period	<Year>–<Year>
◦ Accquired qualifications	**<Title>**
◦ Institute	<Name of educational institution>
◦ Principal subjects	<List of the major subjects>
◦ Minor subjects	<List of minor subjects>
◦ Grade	Evarage grade <overall evarage grade>
▹ Period	<Year>–<Year>
◦ Accquired qualifications	<Title>
◦ Graduate school	<Name of the school>

RESEARCH

▹ Diploma thesis	"<Title of the diploma thesis>" — <Institute>
▹ Seminar paper	"<Title of the seminar paper>" — <Institute>

FIGURA 5: Pacchetto ecv.

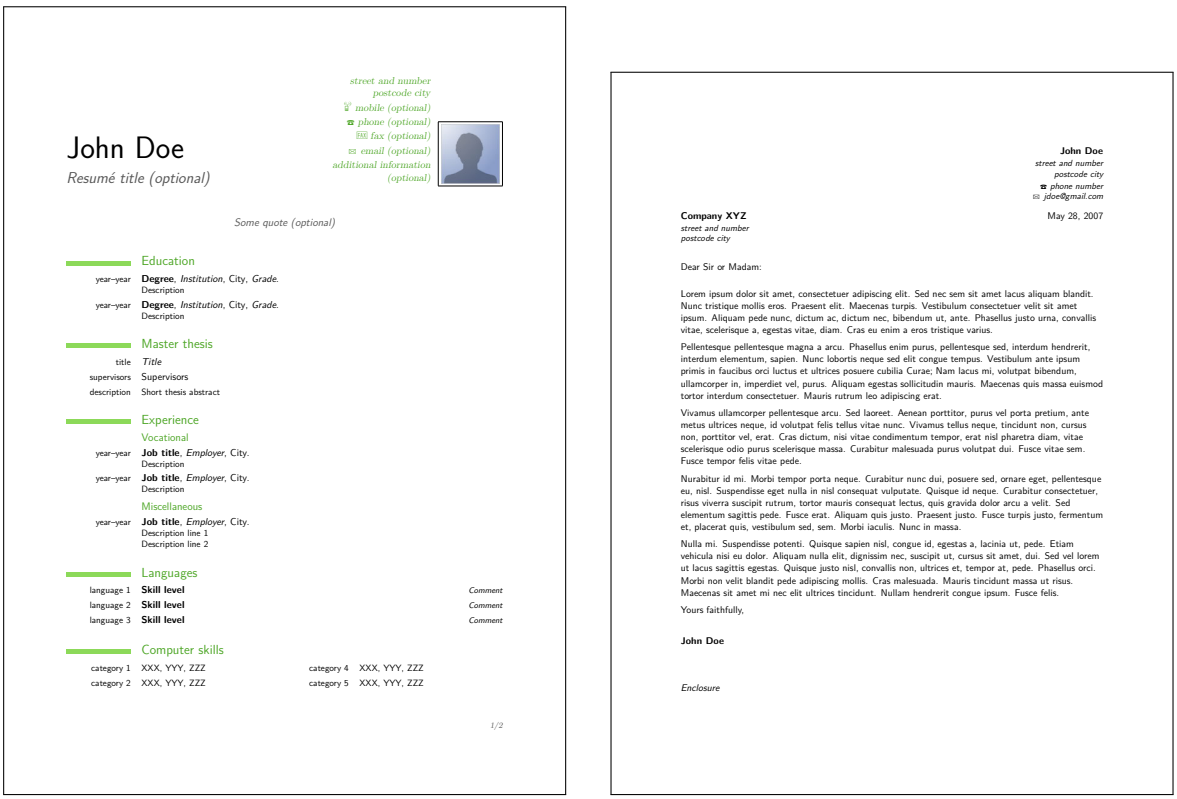


FIGURA 6: Classe moderncv.

il cui utilizzo resta consigliato. Sufficiente la documentazione, che si limita ad una descrizione di base dei comandi forniti.

4.6 moderncv

moderncv, classe scritta da Xavier Danaux nel 2007, è probabilmente una delle migliori soluzioni attualmente disponibili per realizzare un curriculum vitæ in LaTeX. Caratterizzata da un layout tipograficamente molto chiaro ed accattivante, possiede numerosissimi comandi dedicati che permettono di soddisfare anche utenti molto esigenti e con profili professionali particolari. I *template*, forniti come esempio, sono chiari, ben strutturati e ricchi di commenti. Il neofita dovrà solamente sostituire con i propri i dati fittizi presenti all'interno dell'esempio, per disporre in pochi minuti di un gradevole curriculum vitæ.

La classe permette di scegliere tra due varianti, la casual e la classic, differenti solo nel posizionamento dell'eventuale fotografia e delle informazioni personali; per entrambe è inoltre possibile personalizzare il colore degli elementi di separazione. Se si pensa di mettere a disposizione il proprio curriculum sul web si apprezzerà senz'altro il fatto che sia stata già prevista la possibilità di personalizzare le impostazioni dell'`hypersetup` che, inserendo dei metamarcatori all'interno del `.pdf`, facilita il trasferimento delle informazioni ai motori di ricerca. Il *template* è già collegato ad un file `.bib` per inserire l'elenco delle pubblicazioni.

Unico caso nel suo genere, moderncv mette anche a disposizione un *template* per redigere la lettera di presentazione e motivazione, il cui layout richiama nelle linee essenziali lo stile del curriculum.

5 Soluzioni obsolete

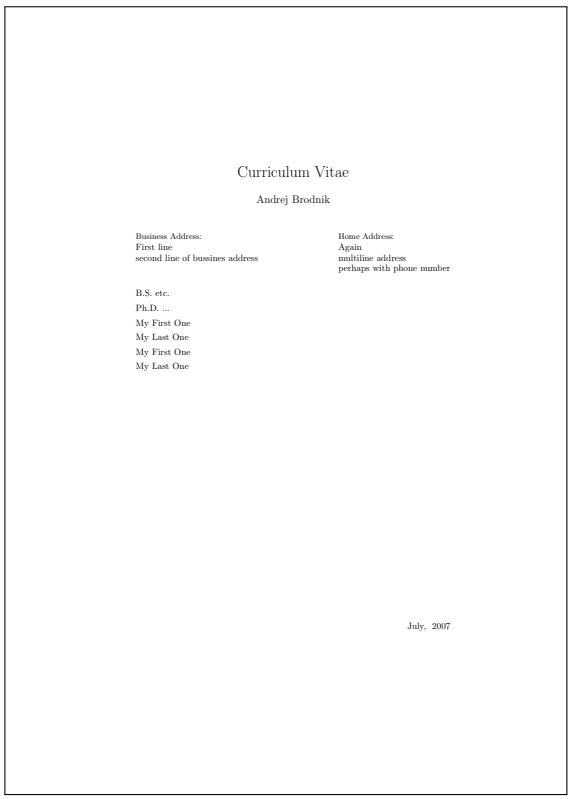
5.1 vita

Scritta nel 1995 da Andrej Brodnik, vita è una classe estremamente semplice ed ormai obsoleta. La struttura si compone semplicemente in una serie di elenchi puntati personalizzati (Degrees, Publications, etc.) che coprono i principali campi di attività scientifica. Il risultato lascia poco spazio alla personalizzazione e davvero nessuno spazio al buon gusto estetico.

Non viene fornita alcuna guida col pacchetto: le istruzioni ed un *template* si trovano direttamente all'interno del file `.cls`.

5.2 CV

Scritto da Gilles Marcou ed Antonio Pereira nel 2004, CV è un pacchetto che utilizza codice ormai datato e che lavora come una semplice estensione della classe article. Non prevedendo comandi dedicati, il codice resta molto articolato. Per suddividere il curriculum vitæ si deve ricorrere a numerosi ambienti tra cui itemize, table, minipage, etc. che vanificano *de facto* qualunque semplificazione derivante dall'utilizzo di un pacchetto dedicato. Il giudizio complessivo è negativo in quanto, se da una parte non viene offerta nessuna *feature* minimamente accattivante per l'utente esperto, dall'altra



Curriculum Vitae

Andrej Brodnik

Business Address:
First line
second line of bussines address

Home Address:
Again
multiline address
perhaps with phone number

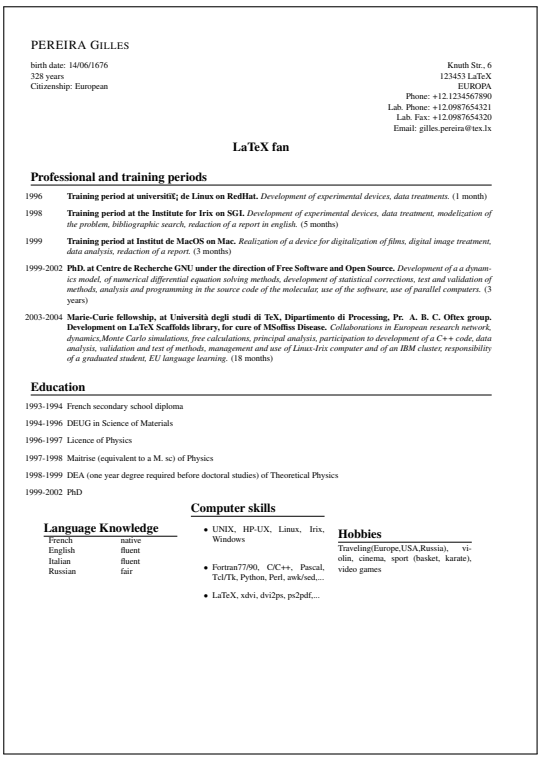
B.S. etc.

Ph.D. ...

My First One
My Last One
My First One
My Last One

July, 2007

FIGURA 7: Classe vita.



PEREIRA GILLES

birth date: 14/06/1676
328 years
Citizenship: European

Knuth Str., 6
123453 LaTeX
EUROPA
Phone: +12.1234567890
Lab. Phone: +12.0987654321
Lab. Fax: +12.0987654320
Email: gilles.pereira@tex.lx

LaTeX fan

Professional and training periods

1996 **Training period at universitiŋ de Linux on RedHat.** *Development of experimental devices, data treatments.* (1 month)

1998 **Training period at the Institute for Irix on SGI.** *Development of experimental devices, data treatment, modelization of the problem, bibliographic search, redaction of a report in english.* (5 months)

1999 **Training period at Institut de MacOS on Mac.** *Realization of a device for digitalization of films, digital image treatment, data analysis, redaction of a report.* (3 months)

1999-2002 **PhD. at Centre de Recherche GNU under the direction of Free Software and Open Source.** *Development of a a dynamics model, of numerical differential equation solving methods, development of statistical corrections, test and validation of methods, analysis and programming in the source code of the molecular, use of the software, use of parallel computers.* (3 years)

2003-2004 **Marie-Curie fellowship, at Università degli studi di TeX, Dipartimento di Processing, Pr. A. B. C. Oftex group. Development on LaTeX Scaffolds library, for cure of MSoftiss Disease.** *Collaborations in European research network, dynamics,Monte Carlo simulations, free calculations, principal analysis, participation to development of a C++ code, data analysis, validation and test of methods, management and use of Linux-Irix computer and of an IBM cluster, responsibility of a graduated student, EU language learning.* (18 months)

Education

1993-1994 French secondary school diploma
1994-1996 DEUG in Science of Materials
1996-1997 Licence of Physics
1997-1998 Maitrise (equivalent to a M. sc) of Physics
1998-1999 DEA (one year degree required before doctoral studies) of Theoretical Physics
1999-2002 PhD

Language Knowledge

French	native
English	fluent
Italian	fluent
Russian	fair

Computer skills

- UNIX, HP-UX, Linux, Irix, Windows
- Fortran77/90, C/C++, Pascal, Tcl/Tk, Python, Perl, awk/sed,...
- LaTeX, xdvi, dvi2ps, ps2pdf,...

Hobbies

Traveling(Europe,USA,Russia), violin, cinema, sport (basket, karate), video games

FIGURA 8: Pacchetto CV.

non mette a suo agio neanche il neofita, che si troverà intrappolato in una moltitudine di ambienti e di parametri facendo apparire il tutto molto più complicato di quanto in effetti non sia veramente. Il *template* fornito con il pacchetto riporta anche alcuni errori elementari (i simboli di `_` e di `@` non sono preceduti dal necessario \, LaTeX invece di LaTeX e così via).

Il risultato finale resta comunque un buon esempio di come *non* vada fatto un curriculum vitæ: disordinato e lontano dai più elementari canoni tipografici.

5.3 resume

resume è un piccolo pacchetto implementato nel lontano 1987 da Stephen Gildea. Esso permette di scrivere un *résumé* abbastanza semplice e, tutto sommato, chiaro. Il pacchetto lavora come un'estensione della classe article fornendo numerosi comandi dedicati che ne rendono immediato l'utilizzo. Nonostante vi sia la presenza di codice bandito già da molto tempo (viene ancóra usato `\bf` invece di `\textbf`) e di qualche grosso limite in termini di estetica, resta comunque una soluzione ancóra valida e semplice da usare.

5.4 res

res è una classe LaTeX 2_ε scritta da Michael DeCorte nel 1988 per la stesura di *résumé*. Il codice risente ormai troppo degli anni e se ne sconsiglia pertanto l'utilizzo.

6 Personalizzare lo stile

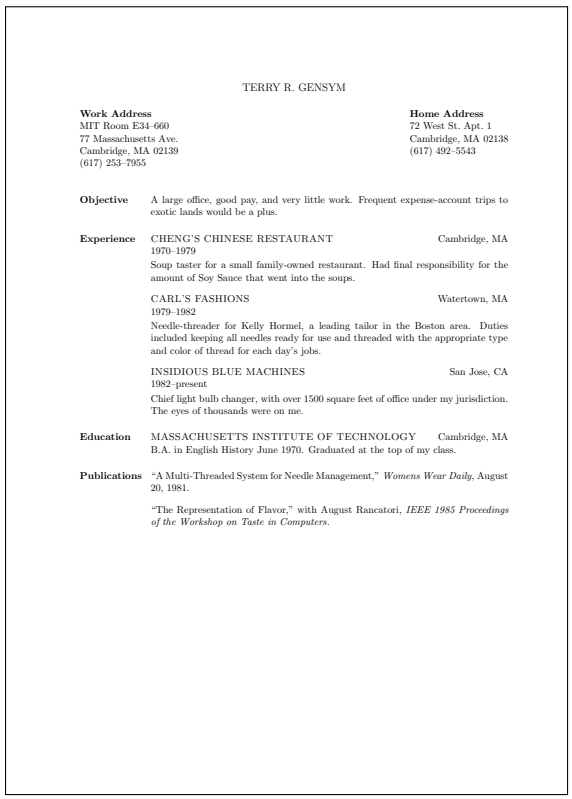
Le classi e i pacchetti disponibili prevedono delle strutture logiche e grafiche più o meno rigide, alle quali l'utente deve adattare il proprio profilo. Talvolta esse possono risultare inadatte a soddisfare le esigenze dell'utente. A differenza di un articolo o di una relazione, infatti, il curriculum vitæ è un documento in cui la personalizzazione della grafica può avere un peso rilevante nel contesto in cui viene proposto. Appare ovvio, quindi, che molte persone possano sentirsi tentate a svincolarsi dalle soluzioni già disponibili per svilupparne di proprie, 'cucite' ad hoc sui propri gusti e sulle proprie esigenze.

La personalizzazione può essere fatta partendo da una classe base e modificandone i comandi o aggiungendo nuovi pacchetti fino a raggiungere gli effetti desiderati; a tal proposito occorre ricordare che, generalmente, questo genere di applicazioni non ha un codice particolarmente complicato e che leggère modifiche sono anche alla portata di un utente non particolarmente esperto.

Se si hanno le competenze tecniche e le idee molto chiare su cosa si vuole ottenere, la soluzione migliore resta comunque quella di partire da zero e creare un nuovo pacchetto o una nuova classe (magari da condividere su CTAN).

7 Conclusioni

Nonostante LaTeX offra una serie di buone soluzioni e facilmente accessibili anche per utenti poco esperti, la scelta da adottare resta comunque un



TERRY R. GENSYM

Work Address
MIT Room E34–660
77 Massachusetts Ave.
Cambridge, MA 02139
(617) 253–7955

Home Address
72 West St. Apt. 1
Cambridge, MA 02138
(617) 492–5543

Objective A large office, good pay, and very little work. Frequent expense-account trips to exotic lands would be a plus.

Experience
CHENG'S CHINESE RESTAURANT — Cambridge, MA
1970–1979
Soup taster for a small family-owned restaurant. Had final responsibility for the amount of Soy Sauce that went into the soups.

CARL'S FASHIONS — Watertown, MA
1979–1982
Needle-threader for Kelly Hormel, a leading tailor in the Boston area. Duties included keeping all needles ready for use and threaded with the appropriate type and color of thread for each day's jobs.

INSIDIOUS BLUE MACHINES — San Jose, CA
1982–present
Chief light bulb changer, with over 1500 square feet of office under my jurisdiction. The eyes of thousands were on me.

Education MASSACHUSETTS INSTITUTE OF TECHNOLOGY — Cambridge, MA
B.A. in English History June 1970. Graduated at the top of my class.

Publications "A Multi-Threaded System for Needle Management," *Womens Wear Daily*, August 20, 1981.

"The Representation of Flavor," with August Rancatori, *IEEE 1985 Proceedings of the Workshop on Taste in Computers*.

FIGURA 9: Pacchetto resume.

elemento estremamente soggettivo e legato alle esigenze personali. Se un modello standard, come ad esempio quello suggerito dalla Comunità Europea, non consentisse alcun tipo di personalizzazione, probabilmente l'utente avanzato sentirebbe il bisogno di apportare modifiche ai modelli esistenti, fino al caso estremo dell'implementazione di una nuova classe *ad hoc*. A tal proposito si tiene a segnalare che le due classi moderncv e eurocv utilizzano una sintassi ed un'architettura del codice più moderna e pienamente compatibile con LaTeX 2ε, prestandosi meglio a svariate personalizzazioni.

A Uso del plurale latino in italiano

Si riporta di seguito un estratto di SETTI (2002) che spiega come mai il plurale di *curriculum* sia *curricula*.

> La parola latina *curriculum* entra in italiano attraverso la locuzione invariabile *curriculum vitae* 'corso della vita in breve' (prima documentazione in italiano 1892) e solo successivamente si trova la forma singola *curriculum* (documentato in italiano dal 1941). Anche la parola singola si afferma nell'uso nella forma invariabile, e nei vocabolari italiani è appunto annotata come sostantivo maschile invariabile. Sappiamo però che questo sostantivo in latino apparteneva al genere neutro per cui entrando nell'italiano ha subìto un passaggio di genere, venendo assimilato ai nomi maschili. Proprio questo passaggio di genere produce problemi e incertezze nel momento che si consideri la forma plurale; infatti, anche se esiste la forma adattata in italiano *curricolo* con il suo plurale regolare *curricoli*, il termine originario latino continua ad essere quello preferito e più utilizzato. Il plurale di *curriculum* è senza dubbio *curricula*, ma questa forma viene usata con qualche resistenza perché, se la forma singolare è talmente diffusa che si suppone sia conosciuta da tutti, il plurale del neutro latino con la tipica terminazione in -a è più lontano dal sistema morfologico dell'italiano e presuppone la conoscenza almeno delle nozioni elementari del latino. Si può quindi scegliere tra la parola italianizzata *curricolo* con plurale *curricoli* o la forma originaria latina *curriculum* con il suo plurale *curricula*.
>
> (SETTI, 2002)

Riferimenti bibliografici

ADANI, L. (2001). *Scrivere il curriculum, 31 modi per essere efficaci*. Etas libri, Milano, Italy.

— (2002). *Il curriculum vincente. Per trovare o cambiare lavoro*. Etas, Milano, Italy.

AMADORI, A. e DE GIULI, A. (1993). *Come farsi assumere. Dal curriculum al colloquio di selezione*. Sperling Paperback.

ANNOVAZZI, L. (1998). *Il curriculum vitæ perfetto*. De Vecchi, Milano, Italy, 1ª edizione.

ANTHONY, R. e ROE, G. (1998). *The Curriculum Vitæ Handbook: How to Present and Promote Your Academic Career*. Rudi Publishing, San Francisco, USA, 2ª edizione.

BARBASIO, L. e PICCARDO, N. (2003). *Curriculum vitæ in inglese*. De Vecchi, Milano, Italy, 1ª edizione.

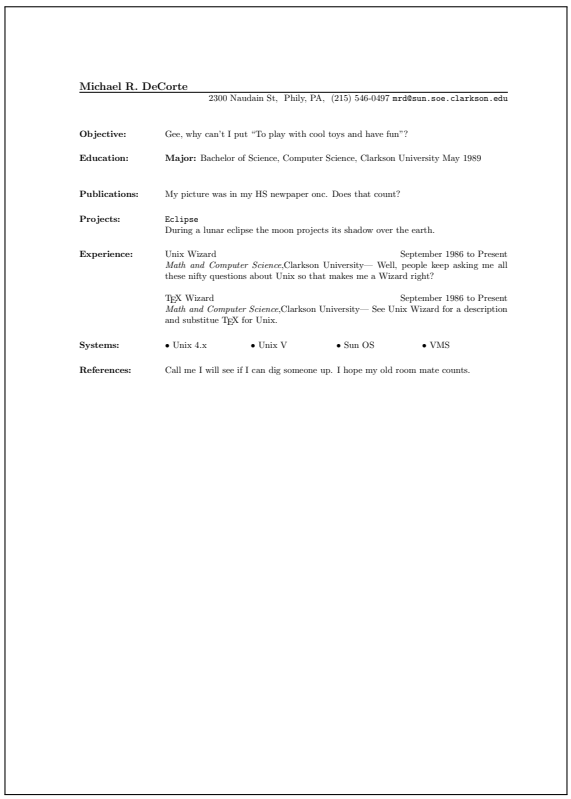
BAUDOIN, M. (1998). *Impara LaTeX (...e mettilo da parte)*. `http://www.mat.uniroma1.it/centro-calcolo/manuali/impara_latex.pdf`.

BECCARI, C. (2007). *Introduzione all'arte della composizione tipografica*. `http://www.guit.sssup.it/downloads/GuidaGuIT.pdf`.

BIGLINO, R. (1993). *Le regole del curriculum vitæ efficace. Come farsi convocare e ottenere la posizione che interessa*. De Vecchi, Milano, Italy.

BOSCHI, P. e LEPORE, M. (2001). *Fare il curriculum. La chiave per un lavoro nuovo e congeniale*. Giunti Demetra, Firenze, Italy, 1ª edizione.

BRUNI, F., FORNASIERO, S. e TAMIOZZO GOLDMANN, S. (1997). *Manuale di scrittura professionale. Dal curriculum vitæ ai documenti aziendali*. Zanichelli, Bologna, Italy.



Michael R. DeCorte
2300 Naudain St, Phily, PA, (215) 546-0497 mrd@sun.soe.clarkson.edu

Objective: Gee, why can't I put "To play with cool toys and have fun"?

Education: **Major:** Bachelor of Science, Computer Science, Clarkson University May 1989

Publications: My picture was in my HS newpaper onc. Does that count?

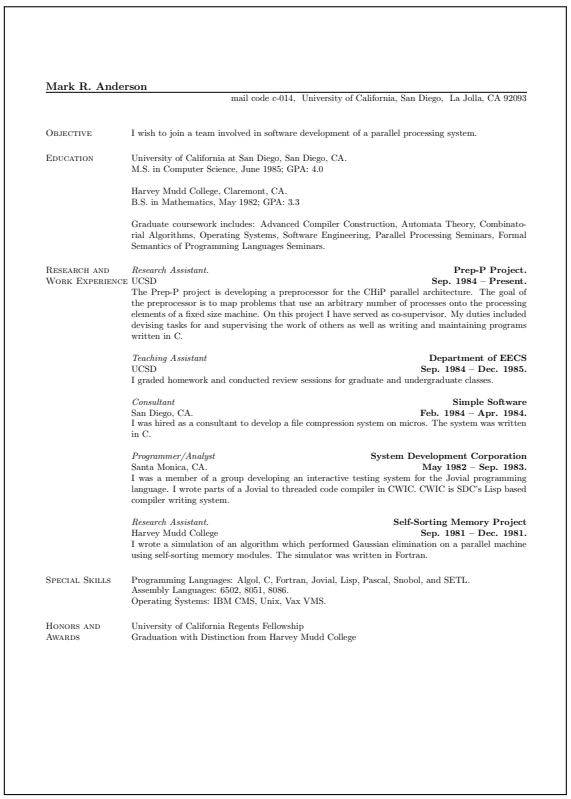
Projects: Eclipse
During a lunar eclipse the moon projects its shadow over the earth.

Experience: Unix Wizard September 1986 to Present
Math and Computer Science,Clarkson University— Well, people keep asking me all these nifty questions about Unix so that makes me a Wizard right?

TEX Wizard September 1986 to Present
Math and Computer Science,Clarkson University— See Unix Wizard for a description and substitue TEX for Unix.

Systems: • Unix 4.x • Unix V • Sun OS • VMS

References: Call me I will see if I can dig someone up. I hope my old room mate counts.



Mark R. Anderson
mail code c-014, University of California, San Diego, La Jolla, CA 92093

OBJECTIVE I wish to join a team involved in software development of a parallel processing system.

EDUCATION University of California at San Diego, San Diego, CA.
M.S. in Computer Science, June 1985; GPA: 4.0

Harvey Mudd College, Claremont, CA.
B.S. in Mathematics, May 1982; GPA: 3.3

Graduate coursework includes: Advanced Compiler Construction, Automata Theory, Combinatorial Algorithms, Operating Systems, Software Engineering, Parallel Processing Seminars, Formal Semantics of Programming Languages Seminars.

RESEARCH AND WORK EXPERIENCE *Research Assistant.* **Prep-P Project.**
UCSD **Sep. 1984 – Present.**
The Prep-P project is developing a preprocessor for the CHiP parallel architecture. The goal of the preprocessor is to map problems that use an arbitrary number of processes onto the processing elements of a fixed size machine. On this project I have served as co-supervisor. My duties included devising tasks for and supervising the work of others as well as writing and maintaining programs written in C.

Teaching Assistant **Department of EECS**
UCSD **Sep. 1984 – Dec. 1985.**
I graded homework and conducted review sessions for graduate and undergraduate classes.

Consultant **Simple Software**
San Diego, CA. **Feb. 1984 – Apr. 1984.**
I was hired as a consultant to develop a file compression system on micros. The system was written in C.

Programmer/Analyst **System Development Corporation**
Santa Monica, CA. **May 1982 – Sep. 1983.**
I was a member of a group developing an interactive testing system for the Jovial programming language. I wrote parts of a Jovial to threaded code compiler in CWIC. CWIC is SDC's Lisp based compiler writing system.

Research Assistant. **Self-Sorting Memory Project**
Harvey Mudd College **Sep. 1981 – Dec. 1981.**
I wrote a simulation of an algorithm which performed Gaussian elimination on a parallel machine using self-sorting memory modules. The simulator was written in Fortran.

SPECIAL SKILLS Programming Languages: Algol, C, Fortran, Jovial, Lisp, Pascal, Snobol, and SETL.
Assembly Languages: 6502, 8051, 8086.
Operating Systems: IBM CMS, Unix, Vax VMS.

HONORS AND AWARDS University of California Regents Fellowship
Graduation with Distinction from Harvey Mudd College

FIGURA 10: Classe res.

BURATTO, F. (2007). *Curriculum atipico di un trentenne tipico*. Marsilio, Venezia, Italy, 1ª edizione.

CAMA, L. (1996). *Trovare lavoro. Strategie e metodi. Quali aziende contattare, come preparare il curriculum, come affrontare il colloquio*. Maggioli, 1ª edizione.

CATALANI, A. (1998). *Il curriculum per convincere. Un libro per trovare non solo un lavoro, ma il proprio lavoro*. Bompiani, Milano, Italy, 1ª edizione.

COMUNITÀ EUROPEA (2003). *Istruzioni per l'uso del curriculum vitæ Europass*. http://europass.cedefop.europa.eu/.

CONSUL, E. e DI FRESCO, A. (2003). *Uno straccio di curriculum*. Il Sole 24 Ore, Milano, Italy.

CRIMINI, P. e GIUSTI, E. (2002). *Come scrivere il proprio curriculum. Le regole d'oro per trovare il lavoro giusto*. Franco Angeli, Milano, Italy.

DEMI, B. e SANTONOCITO, R. (1997). *Il tuo curriculum vitæ: come scriverlo e presentarlo: i profili emergenti e quelli più richiesti*. Il Sole 24 Ore Libri, Milano, Italy.

— (2004). *Il tuo curriculum vitæ. Come scriverlo e presentarlo*. Il Sole 24 Ore Norme & Tributi, Milano, Italy.

DEVOTO, G. e OLI, G. C. (2008). *Il Devoto-Oli. Vocabolario della lingua italiana 2008*. Le Monnier.

DURO, A. (a cura di) (1986). *Vocabolario della lingua italiana*. Isttituto della Enciclopedia Italiana fondata da Giovanni Treccani, Roma, Italy.

FIORINI, M. (1995). *Il tuo C.V. Come scrivere il curriculum e a chi spedirlo*. Sonda, Casale Monferrato, Italy.

GOOSSENS, M., MITTELBACH, F. e SAMARIN, A. (1995). *The LaTeX Companion*. Addison-Wesley.

JACKSON, A. L. e GECKEIS, C. K. (2003). *How to Prepare Your Curriculum Vitæ*. McGraw-Hill, New York, USA, 3ª edizione.

KOPKA, H. e DALY, P. (1995). *A Guide to LaTeX – Document Preparation for Beginners and Advanced Users*. Addison-Wesley.

MEYER, H. (1991). *Introduzione alla metodologia del curriculum*. Armando, Roma, Italy.

OETIKER, T., PARTL, H., HYNA, I. e SCHLEGL, E. (2000). *Una (mica tanto) breve introduzione a LaTeX 2ε*. http://www.ctan.org/tex-archive/info/lshort/italian/itlshort.pdf.

PASSERINI, W. (1999). *99 modi per scrivere un curriculum*. Zelig, 1ª edizione.

Patota, G. (a cura di) (2006). *Il grande dizionario Garzanti della lingua italiana 2006*. Garzanti Linguistica, Milano, Italy.

Rustico, M. (2000). *Curriculum vitæ e lettera di accompagnamento*. De Vecchi, Milano, Italy.

Setti, R. (a cura di) (2002). *Consulenza Linguistica*. Accademia della Crusca. `http://www.accademiadellacrusca.it/faq/`.

The Tutorial Team (2000). *On-line Tutorial on LaTeX*. Indian TeX Users Group. `http://www.tug.org.in/tutorials.html`.

Vascon, M. (2000). *Come si fa un curriculum per conquistare un lavoro*. Giunti Demetra, Firenze, Italy.

Zingarelli, N. (a cura di) (1984). *Vocabolario della lingua italiana*. Zanichelli, Bologna, Italy, 11ª edizione.

▷ Lapo F. Mori
Dipartimento di Ingegneria Meccanica, Nucleare e della Produzione
Università di Pisa
56126 Pisa
`lapo.mori@studenti.ing.unipi.it`

▷ Maurizio W. Himmelmann
Ufficio Statistica
Scuola Superiore Sant'Anna
Pisa
`himmel@sssup.it`

La progettazione di un'opera di consultazione: l'edizione del *Prontuario dell'ingegnere* con LaTeX

Claudio Beccari, Andrea Guadagni

Sommario

Nel presente articolo viene viene descritto l'intero percorso di progettazione di un'opera di consultazione, il *Prontuario dell'ingegnere*, che oltre ad avere le caratteristiche di qualsiasi altra opera di consultazione, fa largo uso del linguaggio matematico e contiene moltissime figure.

Il progetto, realizzato nei primi anni '90, comportava una ulteriore particolarità: dal medesimo materiale si voleva ricavare sia un libro tradizionale sia un testo da consultare direttamente sullo schermo dell'elaboratore.

Abstract

The production of the *Enginer's quick reference book* with LaTeX describes the whole process of designing a reference book; as a reference work it shares the characteristics of any other such book, but it uses also a large amount of mathematics and contains a large number of graphics insertions.

Moreover the publisher wanted to distribute the work both as a regular bound book and as a computer file to be read directly on the computer screen.

1 Introduzione

Nella prima metà degli anni '90 Andrea Guadagni (AG) chiese a Claudio Beccari (CB) di collaborare assieme per produrre un manuale di rapida consultazione, decisamente più snello dello storico ma ormai mastodontico *Manuale dell'ingegnere* del Colombo Colombo (2003), da pubblicare sempre presso Hoepli. L'idea era quella di un prontuario formato da collezioni di schede, ognuna su un argomento particolare, con il minimo di matematica pertinente, possibilmente senza rinvii ad altre schede, così che ognuna fosse 'autoconsistente'.

Come curatore del Manuale dell'ingegnere AG aveva tutta l'esperienza per cercare i collaboratori, per organizzare il materiale, per gestire l'intera operazione da un punto di vista editoriale. Alla fine si occupò anche dell'editing vero e proprio.

Al contrario CB aveva già qualche esperienza per la progettazione in LaTeX del formato delle pagine e del *layout*, ma quella del prontuario era una sfida nuova che richiedeva una forte interazione fra AG e CB.

2 Usare LaTeX negli anni '90

Conviene innanzi tutto accennare ad alcune questioni di carattere informatico che sono ormai superate, ma che allora costituirono un problema; queste difficoltà infatti sono di interesse generale per qualsiasi lavoro svolto in collaborazione fra più autori. Inoltre, anche se con il passare degli anni il progresso informatico elimina vecchie incompatibilità, spesso se ne generano di nuove e altrettanto problematiche. È bene che gli autori che desiderano collaborare alla scrittura di un'opera ne siano al corrente e sappiano come superare le difficoltà che possono trovare sul proprio cammino.

Uno dei problemi era dal fatto che CB lavorava allora solo con un PC DOS; Windows 95 era appena uscito, ma ancora non era sufficientemente diffuso sui PC preesistenti; Windows 3.1 era ancora troppo rudimentale per gestire questo tipo di lavoro, soprattutto perché possedeva una interfaccia grafica per l'utente che consumava quasi tutta la memoria RAM dei PC allora in esercizio. CB inoltre lavorava ancora con LaTeX 209, benché alla fine del lavoro fosse già disponibile LaTeX 2_ε. Usava un editor ASCII e la finestra comandi per lanciare le varie fasi di compilazione.

Al contrario AG lavorava con un Macintosh, il cui sistema operativo era assai diverso dal semplice DOS, se non altro per la diversa modalità con cui venivano indicati i percorsi dei vari file; ovviamente disponeva di una bellissima interfaccia grafica, ma ai fini del progetto in esame la cosa era irrilevante. Per quanto riguarda TeX, AG usava Textures; questo era (ed è tuttora) un programma commerciale della BlueSky Research che funzionava contemporaneamente da *editor* TeX e da *previewer* sincrono, nel senso che teneva aperte due finestre, in una delle quali si componeva il solito sorgente `.tex` e nell'altra si vedeva direttamente il risultato della composizione. La finestra di composizione veniva aggiornata in modo sincrono via via che si immetteva testo nella finestra ASCII del file sorgente.

Oggi tutto è più semplice, da questo punto di vista; il semplice DOS non viene quasi più usato, anche se la finestra comandi di Windows continua a renderlo accessibile, sebbene la stragrande maggioranza degli utenti delle piattaforme Windows per lo più ne ignora l'esistenza; la Apple ha fornito i suoi nuovi Macintosh di un nuovo sistema operativo, il Mac OS X, che è sostanzialmente un sistema

UNIX dotato della ben nota interfaccia grafica di altissimo livello. D'altro canto il Gruppo degli utenti di TEX, TUG, da alcuni anni distribuisce la versione gratuita del sistema TEX mediante la distribuzione TEXlive valida per tutte le piattaforme e le differenze fra i vari sistemi operativi sono ora gestite nello stesso modo, così che i programmi, le collezioni di macro, possano essere sempre più *platform independent*.

Allora però bisognava combattere anche contro i diversi metodi per indicare il percorso assoluto dei file: DOS e UNIX usavano la barra '/', mentre Macintosh usava i due punti anche in posizioni diverse da dove DOS e UNIX usavano la barra. Tanto per riportare un poco della programmazione che fu necessario scrivere per affrontare questo problema, ecco il codice per cambiare localmente la scrittura del percorso assoluto di un file corrispondente al contenuto di una figura.

```
\newif\if@path \@pathfalse
% Queste macro sono prese e adattate dal
% TeXbook di Knuth (pag 375)
\def\check@figurefilepath#1{%
   \@@chkfpM#1:\@@chkfpM\@%
\if@path\else\@@chkfpU#1/\@@chkfpU\@\fi}

\def\@@chkfpM#1:#2#3\@{%
  \ifx\@@chkfpM#2\@pathfalse\else
      \@pathtrue\fi}
\def\@@chkfpU#1/#2#3\@{%
  \ifx\@@chkfpU#2\@pathfalse\else
      \@pathtrue\fi}
%
% Questa e' la macro che eventualmente
% prepende il path al nome del file della
% figura, poi controlla se i separatori
% di folder/directory vanno d'accordo con
% il sistema usato e infine passa il nome
% cosi' modificato alla macro che include
% la figura.
%
\def\parse@figure@filename#1{%
\def\figurefilename{#1}%
\ifx\def@ultpath\empty\else
  \expandafter\check@figurefilepath
  \expandafter{\figurefilename}%
  \if@path\else
     \edef\figurefilename{%
     \def@ultpath\ifMacintosh:%
        \else/\fi\figurefilename}%
  \fi
\fi
\ifMacintosh
  \expandafter\Mac@rename
     \figurefilename/\relax
\else
  \expandafter\UNIX@rename
     \figurefilename:\relax
\fi}
%
% Macro ricorsiva che toglie gli slash e
% mette i due punti
%
\def\Mac@rename#1/#2\relax{%
    \def\@tempA{#2}%
    \ifx\@tempA\empty
        \def\figurefilename{#1}%
    \else
        \def\@tempA{#1:#2}%
        \expandafter\remove@slash
            \@tempA\relax
        \expandafter\Mac@rename
            \figurefilename/\relax
    \fi}
%
% Macro ricorsiva che toglie i due
% punti e mette gli slash
%
\def\UNIX@rename#1:#2\relax{%
  \def\@tempA{#2}%
  \ifx\@tempA\empty
     \def\figurefilename{#1}\else
     \def\@tempA{#1/#2}%
     \expandafter\remove@colon
         \@tempA\relax
     \expandafter\UNIX@rename
        \figurefilename:\relax
  \fi}
%
% Queste macro tolgono lo slash o i due
% punti finali
%
\def\remove@slash#1/\relax{%
   \def\figurefilename{#1}}
\def\remove@colon#1:\relax{%
    \def\figurefilename{#1}}
```

3 L'impostazione grafica del volume

Per il libro 'cartaceo' AG e CB concordarono un formato di layout che impostasse ogni scheda come una sezione a se stante, a livello di `\section`; però ogni scheda doveva cominciare su pagina nuova; nella versione cartacea, dove le pagine si sfogliano voltando il foglio verso sinistra, era necessario che il titolo e la parte scritta comparissero sul recto (pagina dispari), mentre la parte grafica sarebbe apparsa sul verso del foglio precedente.

Nella versione elettronica, invece, dove le pagine scorrono dal basso verso l'alto, ma la lettura procede da sinistra a destra, la scheda avrebbe dovuto occupare una sola schermata, con la parte scritta a sinistra e la parte grafica a destra.

Entrambe le versioni avrebbero avuto le 'unghie' a margine delle pagine; nella versione cartacea sarebbe stato desiderabile intagliare il bordo delle pagine a semicerchio, come si vede talvolta nei libri sacri di grandi dimensioni, oppure nei grandi

volumi di certi dizionari o di certe enciclopedie. In realtà l'operazione avrebbe incrementato il costo in modo notevole, quindi ci si accontentò che sul taglio esterno delle pagine apparisse l'ombra dell'unghia ottenuta stampando l'unghia oltre al margine fisico del foglio, in modo tale che il taglio rendesse visibile all'esterno una piccola parte della zona inchiostrata dell'unghia (vedi più avanti la figura 5). Questa a sua volta, veniva stampata su tutte le pagine delle parti, così che, maneggiando il libro, lo si sarebbe potuto sfogliare alla ricerca di un argomento, semplicemente sfogliando quella parte con le stesse icone sul margine. Queste icone/unghie non erano quindi una semplice macchia di colore o una lettera stampata sul margine, ma un vero e proprio pittogramma che avrebbe richiamato alla mente subito il genere di argomenti trattati nella parte.

La versione informatica avrebbe avuto tutti gli hyperlink necessari, ma per coerenza di design avrebbe avuto anche i pittogrammi a margine, però questa volta sul margine sinistro.

Testatine, parti, schede, indici generale e analitico richiedevano un design particolare, anche perché doveva essere esplicito il nome del redattore della singola scheda e il nome del coordinatore del capitolo; alcune di queste informazioni dovevano poter comparire anche nell'indice generale.

Il file sorgente della versione cartacea e di quella informatica avrebbe dovuto essere lo stesso e LaTeX avrebbe prodotto l'una o l'altra versione specificando una sola opzione nel file principale.

4 Il layout della pagina e la scelta dei font

Il formato della pagina era UNI-A5, cioè 148 mm per 210 mm; la gabbia del testo, senza contare testatine e piedini, era di 116 mm×175 mm. Data la compattezza del testo si decise di comporre in corpo 9/10; rinviando la decisione sulla scelta dei font all'analisi dettagliata dei risultati compositivi. Si decise a posteriori di usare il font Times, che con Textures era (allora) abbastanza facile da usare e da configurare.

La figura 1 mostra chiaramente la disposizione delle due pagine di una scheda; la pagina di sinistra riporta le figure relative alla scheda, la pagina di destra ne riporta il testo. L'unghia, o l'icona che caratterizza la parte (nell'esempio quella relativa all' 'Edilizia') è riportata nella pagina di destra. Il nome della parte è ripetuto nella testatina di sinistra, mentre la testatina di destra riporta il titolo del 'capitolo' che raggruppa un certo numero di schede. Il titolo della scheda è riportato nella pagina di destra; la scheda è suddivisa in piccole sottosezioni il cui titolino è in linea con il testo. L'autore della scheda è riportato nel piedino di destra, mentre il piedino di sinistra indica il titolo dell'opera e l'editore.

Nella figura 2 compaiono pressappoco gli stessi elementi, salvo che unghia, testo e figure occupano la pagina/schermata in ordine inverso rispetto alla versione stampata. Le testatine e i piedini sono diventati elementi della stessa testatina e dello stesso piedino ed hanno collocazioni in bandiera diverse dalla versione stampata. Per il resto non cambia quasi nulla.

La versione stampata ha le pagine in formato A5, mentre la versione elettronica ha le schermate in posizione panoramica (landscape) con le dimensioni di una foglio A4; se si volesse stampare una schermata, bisogna ricordarsi di impostare la disposizione landscape per la stampante e bisogna essere sicuri che essa usi carta A4 o che essa sia in grado di adattare le dimensioni della pagina a quelle della carta.

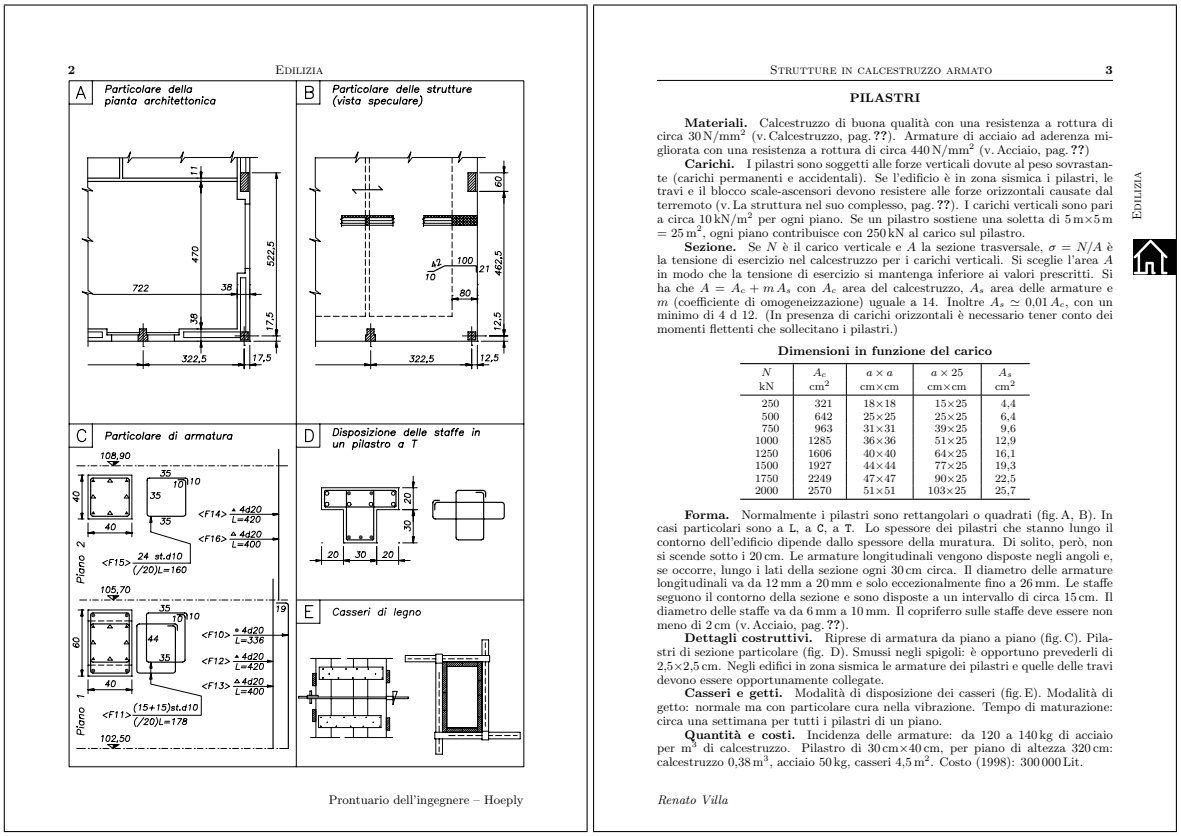
5 I comandi per la composizione delle schede

La progettazione del prontuario in LaTeX richiede chiaramente comandi specifici e soluzioni particolari adeguate ai diversi problemi posti dalla sua composizione tipografica.

Per esempio, la figura di ogni scheda deve essere assemblata separatamente, possibilmente con un editor grafico in modo da disporre di un unico oggetto da includere nel file di uscita. Per quei tempi era assolutamente indispensabile ricorrere alla composizione mediante LaTeX per ottenere il file DVI, per poi trasformarlo in formato PS mediante dvips e, infine, ottenere il formato PDF con l'ulteriore trasformazione mediante ps2pdf. Oggi si otterrebbe lo stesso risultato usando direttamente pdflatex. Tuttavia, una volta predisposte apposite *macro*, anche con i pochi mezzi di allora non era difficile sia per l'autore delle singole schede, sia per il curatore ottenere il risultato desiderato.

Le macro per iniziare una nuova scheda e per decidere dove disporre il materiale grafico rispetto a quello testuale richiedevano solo il titolo della scheda specifica e il nome dell'autore. Lo stesso può dirsi per le macro che iniziavano un nuovo capitolo.

Le macro che iniziavano ogni parte dovevano provvedere anche alla scelta dell'icona specifica, alla sua collocazione a destra nella versione stampata, o a sinistra nella versione elettronica, ma non presentavano difficoltà particolari. Si rese invece necessario introdurre una variazione attivabile con l'opzione `draft`; visto che le bozze erano stampate più volte sia dagli autori sia dal curatore, la quantità di inchiostro da usare per le icone delle unghie sarebbe stata eccessiva e il costo di produzione sarebbe stato corrispondentemente alto. Per le bozze, quindi, si decise di non stampare le unghie, lasciando solo la scrittura del titolo della parte che normalmente apparirebbe sopra la sua icona.



A *Particolare della pianta architettonica*

B *Particolare delle strutture (vista speculare)*

C *Particolare di armatura*

D *Disposizione delle staffe in un pilastro a T*

E *Casseri di legno*

PILASTRI

Materiali. Calcestruzzo di buona qualità con una resistenza a rottura di circa 30 N/mm^2 (v. Calcestruzzo, pag. ??). Armature di acciaio ad aderenza migliorata con una resistenza a rottura di circa 440 N/mm^2 (v. Acciaio, pag. ??)

Carichi. I pilastri sono soggetti alle forze verticali dovute al peso sovrastante (carichi permanenti e accidentali). Se l'edificio è in zona sismica i pilastri, le travi e il blocco scale-ascensori devono resistere alle forze orizzontali causate dal terremoto (v. La struttura nel suo complesso, pag. ??). I carichi verticali sono pari a circa 10 kN/m^2 per ogni piano. Se un pilastro sostiene una soletta di 5 m×5 m = 25 m^2, ogni piano contribuisce con 250 kN al carico sul pilastro.

Sezione. Se N è il carico verticale e A la sezione trasversale, $\sigma = N/A$ è la tensione di esercizio nel calcestruzzo per i carichi verticali. Si sceglie l'area A in modo che la tensione di esercizio si mantenga inferiore ai valori prescritti. Si ha che $A = A_c + m\,A_s$ con A_c area del calcestruzzo, A_s area delle armature e m (coefficiente di omogeneizzazione) uguale a 14. Inoltre $A_s \simeq 0{,}01\,A_c$, con un minimo di 4 d 12. (In presenza di carichi orizzontali è necessario tener conto dei momenti flettenti che sollecitano i pilastri.)

Dimensioni in funzione del carico

N kN	A_c cm^2	$a \times a$ cm×cm	$a \times 25$ cm×cm	A_s cm^2
250	321	18×18	15×25	4,4
500	642	25×25	25×25	6,4
750	963	31×31	39×25	9,6
1000	1285	36×36	51×25	12,9
1250	1606	40×40	64×25	16,1
1500	1927	44×44	77×25	19,3
1750	2249	47×47	90×25	22,5
2000	2570	51×51	103×25	25,7

Forma. Normalmente i pilastri sono rettangolari o quadrati (fig. A, B). In casi particolari sono a L, a C, a T. Lo spessore dei pilastri che stanno lungo il contorno dell'edificio dipende dallo spessore della muratura. Di solito, però, non si scende sotto i 20 cm. Le armature longitudinali vengono disposte negli angoli e, se occorre, lungo i lati della sezione ogni 30 cm circa. Il diametro delle armature longitudinali va da 12 mm a 20 mm e solo eccezionalmente fino a 26 mm. Le staffe seguono il contorno della sezione e sono disposte a un intervallo di circa 15 cm. Il diametro delle staffe va da 6 mm a 10 mm. Il copriferro sulle staffe deve essere non meno di 2 cm (v. Acciaio, pag. ??).

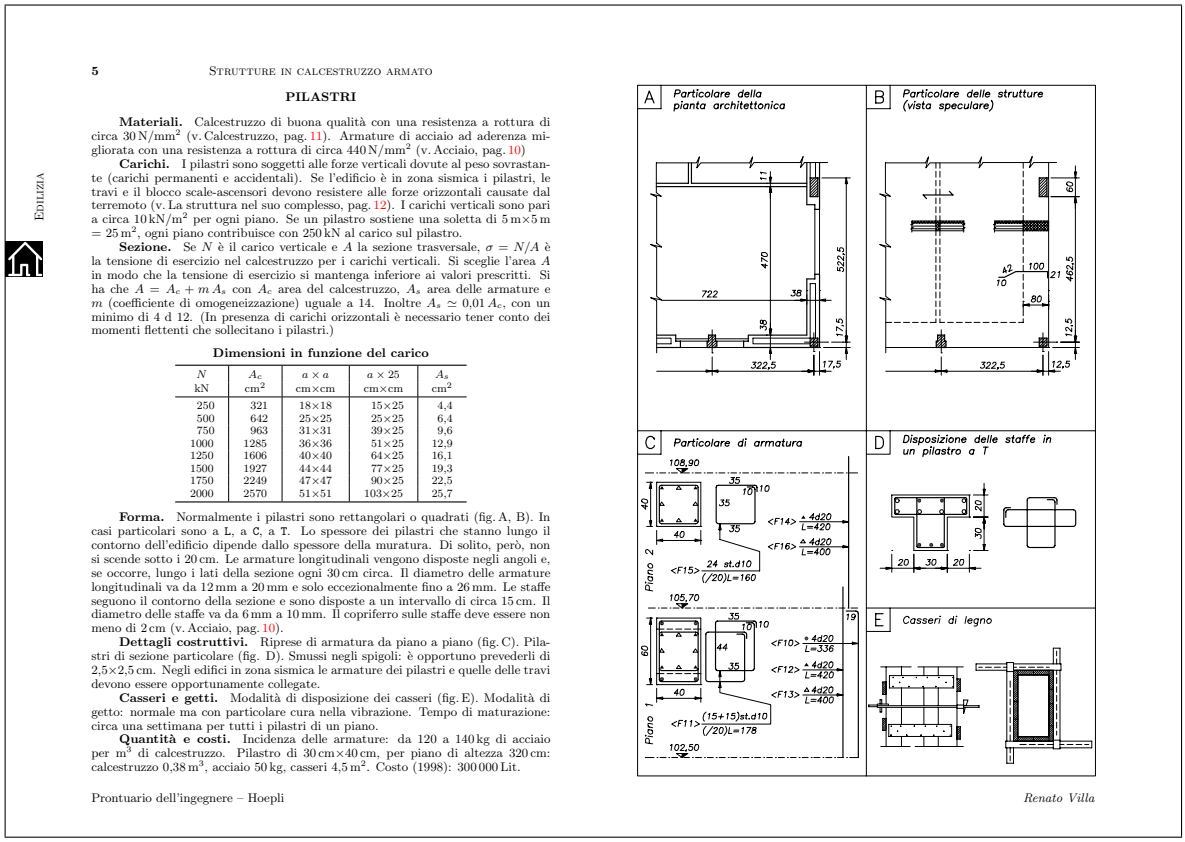
Dettagli costruttivi. Riprese di armatura da piano a piano (fig. C). Pilastri di sezione particolare (fig. D). Smussi negli spigoli: è opportuno prevederli di 2,5×2,5 cm. Negli edifici in zona sismica le armature dei pilastri e quelle delle travi devono essere opportunamente collegate.

Casseri e getti. Modalità di disposizione dei casseri (fig. E). Modalità di getto: normale ma con particolare cura nella vibrazione. Tempo di maturazione: circa una settimana per tutti i pilastri di un piano.

Quantità e costi. Incidenza delle armature: da 120 a 140 kg di acciaio per m^3 di calcestruzzo. Pilastro di 30 cm×40 cm, per piano di altezza 320 cm: calcestruzzo 0,38 m^3, acciaio 50 kg, casseri 4,5 m^2. Costo (1998): 300 000 Lit.

Renato Villa

FIGURA 1: Versione cartacea: il verso e il recto di due pagine successive



PILASTRI

Materiali. Calcestruzzo di buona qualità con una resistenza a rottura di circa 30 N/mm^2 (v. Calcestruzzo, pag. 11). Armature di acciaio ad aderenza migliorata con una resistenza a rottura di circa 440 N/mm^2 (v. Acciaio, pag. 10)

Carichi. I pilastri sono soggetti alle forze verticali dovute al peso sovrastante (carichi permanenti e accidentali). Se l'edificio è in zona sismica i pilastri, le travi e il blocco scale-ascensori devono resistere alle forze orizzontali causate dal terremoto (v. La struttura nel suo complesso, pag. 12). I carichi verticali sono pari a circa 10 kN/m^2 per ogni piano. Se un pilastro sostiene una soletta di 5 m×5 m = 25 m^2, ogni piano contribuisce con 250 kN al carico sul pilastro.

Sezione. Se N è il carico verticale e A la sezione trasversale, $\sigma = N/A$ è la tensione di esercizio nel calcestruzzo per i carichi verticali. Si sceglie l'area A in modo che la tensione di esercizio si mantenga inferiore ai valori prescritti. Si ha che $A = A_c + m\,A_s$ con A_c area del calcestruzzo, A_s area delle armature e m (coefficiente di omogeneizzazione) uguale a 14. Inoltre $A_s \simeq 0{,}01\,A_c$, con un minimo di 4 d 12. (In presenza di carichi orizzontali è necessario tener conto dei momenti flettenti che sollecitano i pilastri.)

Dimensioni in funzione del carico

N kN	A_c cm^2	$a \times a$ cm×cm	$a \times 25$ cm×cm	A_s cm^2
250	321	18×18	15×25	4,4
500	642	25×25	25×25	6,4
750	963	31×31	39×25	9,6
1000	1285	36×36	51×25	12,9
1250	1606	40×40	64×25	16,1
1500	1927	44×44	77×25	19,3
1750	2249	47×47	90×25	22,5
2000	2570	51×51	103×25	25,7

Forma. Normalmente i pilastri sono rettangolari o quadrati (fig. A, B). In casi particolari sono a L, a C, a T. Lo spessore dei pilastri che stanno lungo il contorno dell'edificio dipende dallo spessore della muratura. Di solito, però, non si scende sotto i 20 cm. Le armature longitudinali vengono disposte negli angoli e, se occorre, lungo i lati della sezione ogni 30 cm circa. Il diametro delle armature longitudinali va da 12 mm a 20 mm e solo eccezionalmente fino a 26 mm. Le staffe seguono il contorno della sezione e sono disposte a un intervallo di circa 15 cm. Il diametro delle staffe va da 6 mm a 10 mm. Il copriferro sulle staffe deve essere non meno di 2 cm (v. Acciaio, pag. 10).

Dettagli costruttivi. Riprese di armatura da piano a piano (fig. C). Pilastri di sezione particolare (fig. D). Smussi negli spigoli: è opportuno prevederli di 2,5×2,5 cm. Negli edifici in zona sismica le armature dei pilastri e quelle delle travi devono essere opportunamente collegate.

Casseri e getti. Modalità di disposizione dei casseri (fig. E). Modalità di getto: normale ma con particolare cura nella vibrazione. Tempo di maturazione: circa una settimana per tutti i pilastri di un piano.

Quantità e costi. Incidenza delle armature: da 120 a 140 kg di acciaio per m^3 di calcestruzzo. Pilastro di 30 cm×40 cm, per piano di altezza 320 cm: calcestruzzo 0,38 m^3, acciaio 50 kg, casseri 4,5 m^2. Costo (1998): 300 000 Lit.

Renato Villa

FIGURA 2: Versione elettronica: la schermata della stessa scheda della figura 1

Il pacchetto babel non era ancora disponibile con le funzionalità che ha oggi; molti utenti di LaTeX allora non sapevano nemmeno che potevano comporre in italiano e/o non sapevano come fare; una delle cose necessarie fu quella di impostare test e messaggi adeguati per garantire l'uso della sillabazione italiana. Il codice era il seguente:

```
\expandafter
    \ifx\csname language\endcsname\relax
\else
  \expandafter
    \ifx\csname l@italian\endcsname\relax
  \typeout{La sillabazione per l'italiano
          non e' definita.^^J
          Usero' la sillabazione di
          default e faro' tanti errori!}%
  \typeout{Verificare che la sillabazione
          per l'italiano sia stata
          caricata^^J
          e che il suo nome sia associato
          al comando \string\l@italian}%
  \language0\righthyphenmin 2\relax
            \lccode`\'=`\'
  \else
    \def\italiano{\language \l@italian
            \righthyphenmin 2\relax
            \lccode`\'=`\'}%
    \italiano
  \fi
\fi
```

6 Le unghie

Per quel che riguarda le icone delle unghie il problema è stato risolto creando appositi font, il cui disegno era in bianco su nero e dove il nero si estendeva a destra del disegno (per le icone della versione stampata), oppure a sinistra (per le icone della versione elettronica), per una larghezza almeno pari a quella dell'icona stessa, in modo da essere sicuri che il sottofondo nero si estendesse ben oltre al taglio esterno della pagina. Questo fatto garantiva la presenza dell'ombreggiatura scura in corrispondenza dell'unghia sul taglio, anche se le pagine non avevano fisicamente l'intaglio semicircolare dell'unghia. Nella figura 3 sono riportate le varie unghie destre e sinistre realizzate inizialmente come font METAFONT, in un secondo tempo trasformati in font outline di tipo 1.

Come mostra la figura 3, le icone sono ricavate in bianco su nero sul margine interno di un grande rettangolo, il cui margine esterno sporge 'fuori dal foglio'; ecco perché quando si taglia la carta, il libro chiuso mostra sul taglio le ombre delle unghie; probabilmente questo fatto non è determinante nell'aiutare il lettore nella navigazione, quanto invece lo è l'icona riportata sul margine della pagina di destra; anche scorrendo rapidamente le pagine è facile rilevare l'icona cercata con grande rapidità.

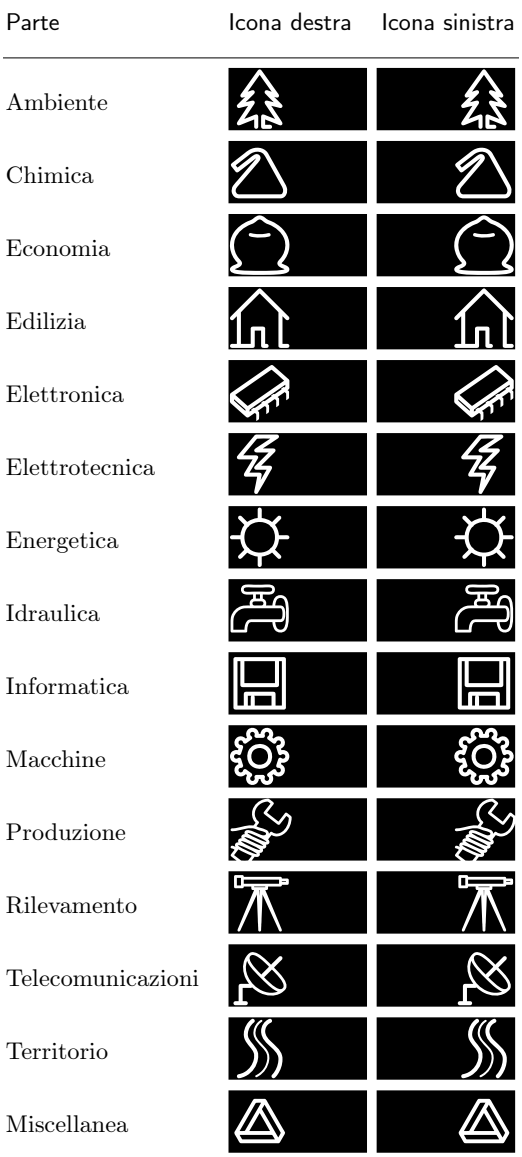


Figura 3: Le icone delle unghie destre e sinistre

Va da sé che le icone di ogni parte vengono progressivamente scalate verso il margine inferiore del foglio, in modo che le icone di una stessa parte appaiono fisse mentre si scorrono le pagine relative.

Anche la versione elettronica, benché abbia le icone sul bordo sinistro e non abbia pagine da far scorrere fra le dita, trae giovamento dalla presenza delle icone quando si fanno scorrere le schermate rapidamente agendo sulle apposite frecce del programma di lettura del file PDF.

7 Hyperlink interni

I diversi elementi della versione elettronica sono completament collegati fra loro con hyperlink, ottenuti tramite il pacchetto hyperref, cosicché la navigazione all'interno del documento procede con semplici click sulle parti evidenziate in colore; anche l'indice generale e l'indice analitico sono collegati con hyperlink. Le schede che rimandano ad altre schede sono anch'esse collegate con hyper-

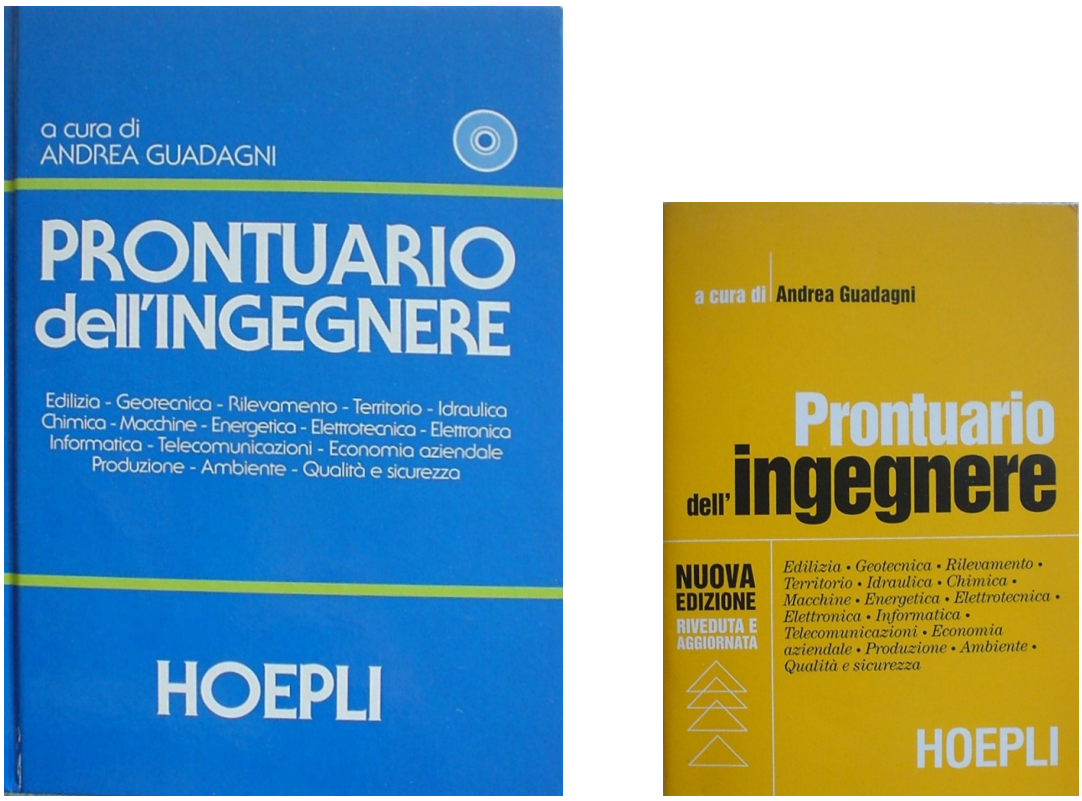


FIGURA 4: I volumi pubblicati; la prima edizione a sinistra (formato A5) e la seconda edizione a destra (formato 120 mm × 170 mm).

link. In questo modo la versione elettronica risulta anche più facilmente consultabile della versione cartacea. Per gli anni '90 si trattava di una novità che evidentemente è stata gradita dagli utenti di quest'opera.

8 Le modifiche della seconda edizione

La seconda edizione GUADAGNI (2003) subì alcune modifiche nell'impostazione grafica rispetto alla prima edizione, riguardanti anche il formato del volume (figura 4). Innanzi tutto la versione cartacea ebbe le due facciate scambiate di posto, come si vede nella figura 6.

Inoltre le unghie furono stampate sempre a sinistra anche nella versione cartacea, ma in entrambe le versioni esse vennero collocate leggermente più vicine al testo; insieme alla figura 6 si osservi anche la schermata della figura 7. In ogni caso le unghie compaiono sul taglio esterno come ombreggiature più scure come si vede nella figura 5.

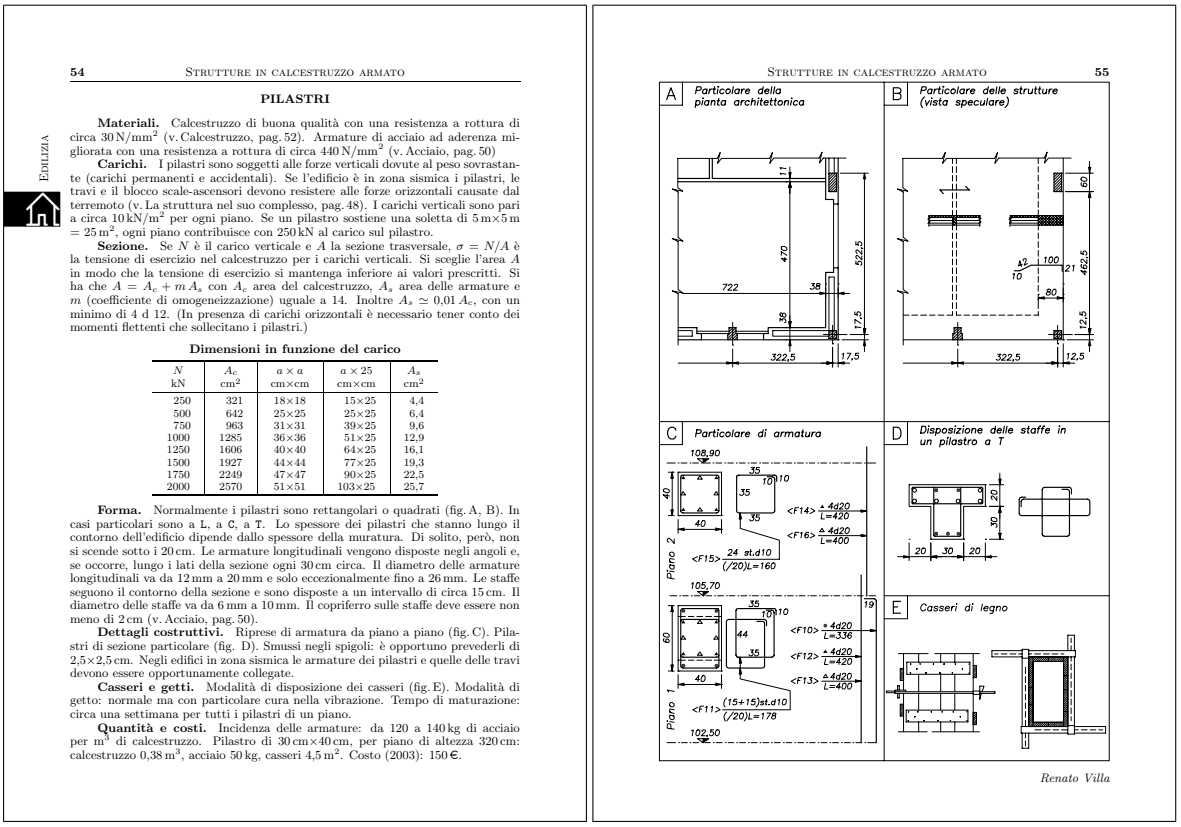
9 L'impostazione editoriale

La Hoepli è nota per i suoi manuali tecnici e, presso gli ingegneri, per il Manuale Colombo: un'opera nata nel 1875 e divenuta ormai enciclopedica, avendo perso da tempo le primitive caratteristiche di sinteticità. Per questo nel *Prontuario dell'ingegnere* si sono voluti condensare solo gli elementi

FIGURA 5: Le unghie sul taglio esterno della seconda edizione

essenziali di ogni specializzazione. Lo scopo è dare un'infarinatura a chi non è specialista di un certo settore, ma vuole capirne i concetti applicativi di base. Questa esigenza è nata dalla collaborazione sempre più frequente fra ingegneri e architetti, ognuno esperto in un campo diverso. Dunque il *Prontuario* va incontro alla necessità di capirsi e di comunicare fra tecnici.

La struttura grafica a schede (una coppia di pagine per ogni argomento) è nata dal desiderio di presentare nel modo più ordinato e schematico possibile i diversi argomenti. Si voleva che anche visivamente l'opera si presentasse come qualcosa di sintetico ed essenziale. Per l'importanza degli schemi grafici e dei disegni tecnici un'intera pagina è stata dedicata alle figure, mentre l'altra pagina contiene il testo. Questa impostazione molto particolare ha influenzato la struttura dei docu-



54 STRUTTURE IN CALCESTRUZZO ARMATO

EDILIZIA

PILASTRI

Materiali. Calcestruzzo di buona qualità con una resistenza a rottura di circa 30 N/mm^2 (v. Calcestruzzo, pag. 52). Armature di acciaio ad aderenza migliorata con una resistenza a rottura di circa 440 N/mm^2 (v. Acciaio, pag. 50)

Carichi. I pilastri sono soggetti alle forze verticali dovute al peso sovrastante (carichi permanenti e accidentali). Se l'edificio è in zona sismica i pilastri, le travi e il blocco scale-ascensori devono resistere alle forze orizzontali causate dal terremoto (v. La struttura nel suo complesso, pag. 48). I carichi verticali sono pari a circa 10 kN/m^2 per ogni piano. Se un pilastro sostiene una soletta di 5 m×5 m = 25 m^2, ogni piano contribuisce con 250 kN al carico sul pilastro.

Sezione. Se N è il carico verticale e A la sezione trasversale, $\sigma = N/A$ è la tensione di esercizio nel calcestruzzo per i carichi verticali. Si sceglie l'area A in modo che la tensione di esercizio si mantenga inferiore ai valori prescritti. Si ha che $A = A_c + m\,A_s$ con A_c area del calcestruzzo, A_s area delle armature e m (coefficiente di omogeneizzazione) uguale a 14. Inoltre $A_s \simeq 0{,}01\,A_c$, con un minimo di 4 d 12. (In presenza di carichi orizzontali è necessario tener conto dei momenti flettenti che sollecitano i pilastri.)

Dimensioni in funzione del carico

N kN	A_c cm^2	$a \times a$ cm×cm	$a \times 25$ cm×cm	A_s cm^2
250	321	18×18	15×25	4,4
500	642	25×25	25×25	6,4
750	963	31×31	39×25	9,6
1000	1285	36×36	51×25	12,9
1250	1606	40×40	64×25	16,1
1500	1927	44×44	77×25	19,3
1750	2249	47×47	90×25	22,5
2000	2570	51×51	103×25	25,7

Forma. Normalmente i pilastri sono rettangolari o quadrati (fig. A, B). In casi particolari sono a L, a C, a T. Lo spessore dei pilastri che stanno lungo il contorno dell'edificio dipende dallo spessore della muratura. Di solito, però, non si scende sotto i 20 cm. Le armature longitudinali vengono disposte negli angoli e, se occorre, lungo i lati della sezione ogni 30 cm circa. Il diametro delle armature longitudinali va da 12 mm a 20 mm e solo eccezionalmente fino a 26 mm. Le staffe seguono il contorno della sezione e sono disposte a un intervallo di circa 15 cm. Il diametro delle staffe va da 6 mm a 10 mm. Il copriferro sulle staffe deve essere non meno di 2 cm (v. Acciaio, pag. 50).

Dettagli costruttivi. Riprese di armatura da piano a piano (fig. C). Pilastri di sezione particolare (fig. D). Smussi negli spigoli: è opportuno prevederli di 2,5×2,5 cm. Negli edifici in zona sismica le armature dei pilastri e quelle delle travi devono essere opportunamente collegate.

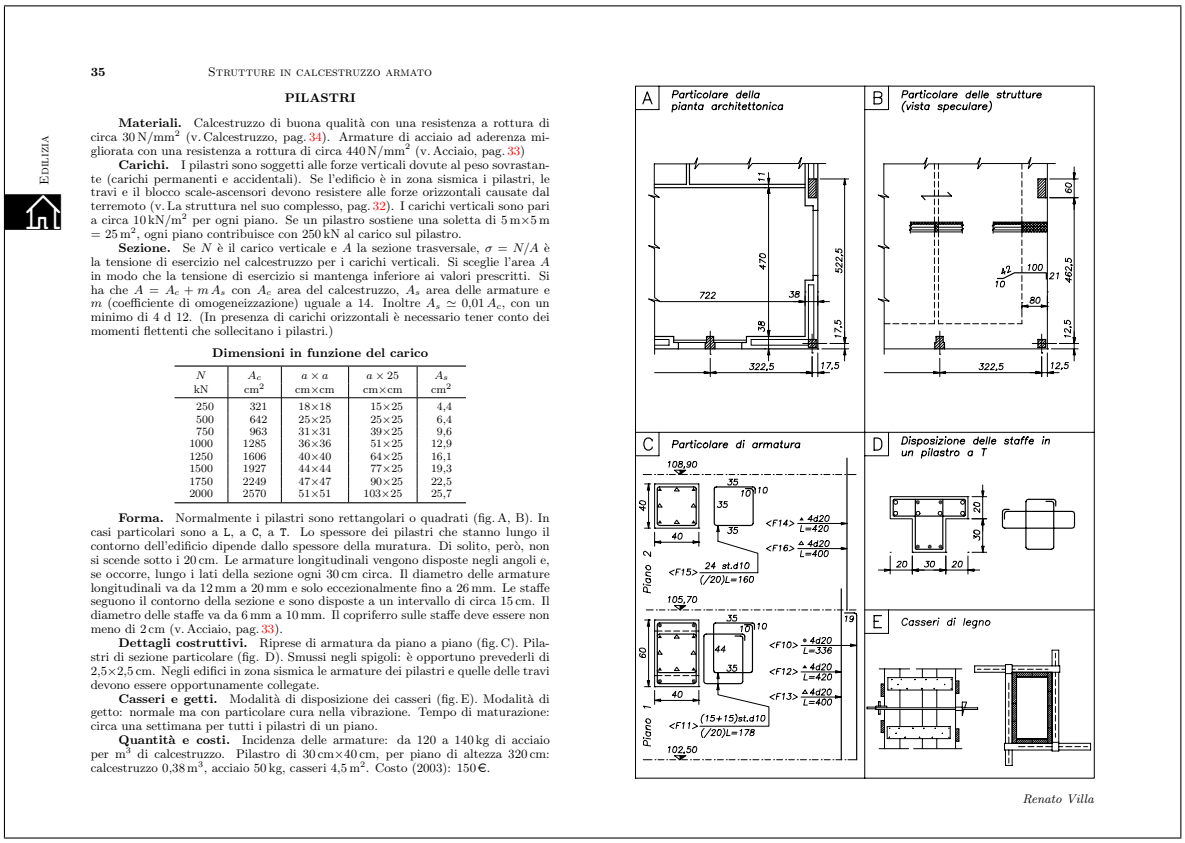
Casseri e getti. Modalità di disposizione dei casseri (fig. E). Modalità di getto: normale ma con particolare cura nella vibrazione. Tempo di maturazione: circa una settimana per tutti i pilastri di un piano.

Quantità e costi. Incidenza delle armature: da 120 a 140 kg di acciaio per m^3 di calcestruzzo. Pilastro di 30 cm×40 cm, per piano di altezza 320 cm: calcestruzzo 0,38 m^3, acciaio 50 kg, casseri 4,5 m^2. Costo (2003): 150 €.

STRUTTURE IN CALCESTRUZZO ARMATO 55

Renato Villa

FIGURA 6: Versione cartacea della seconda edizione: il verso e il recto di due pagine successive



35 STRUTTURE IN CALCESTRUZZO ARMATO

EDILIZIA

PILASTRI

Materiali. Calcestruzzo di buona qualità con una resistenza a rottura di circa 30 N/mm^2 (v. Calcestruzzo, pag. 34). Armature di acciaio ad aderenza migliorata con una resistenza a rottura di circa 440 N/mm^2 (v. Acciaio, pag. 33)

Carichi. I pilastri sono soggetti alle forze verticali dovute al peso sovrastante (carichi permanenti e accidentali). Se l'edificio è in zona sismica i pilastri, le travi e il blocco scale-ascensori devono resistere alle forze orizzontali causate dal terremoto (v. La struttura nel suo complesso, pag. 32). I carichi verticali sono pari a circa 10 kN/m^2 per ogni piano. Se un pilastro sostiene una soletta di 5 m×5 m = 25 m^2, ogni piano contribuisce con 250 kN al carico sul pilastro.

Sezione. Se N è il carico verticale e A la sezione trasversale, $\sigma = N/A$ è la tensione di esercizio nel calcestruzzo per i carichi verticali. Si sceglie l'area A in modo che la tensione di esercizio si mantenga inferiore ai valori prescritti. Si ha che $A = A_c + m\,A_s$ con A_c area del calcestruzzo, A_s area delle armature e m (coefficiente di omogeneizzazione) uguale a 14. Inoltre $A_s \simeq 0{,}01\,A_c$, con un minimo di 4 d 12. (In presenza di carichi orizzontali è necessario tener conto dei momenti flettenti che sollecitano i pilastri.)

Dimensioni in funzione del carico

N kN	A_c cm^2	$a \times a$ cm×cm	$a \times 25$ cm×cm	A_s cm^2
250	321	18×18	15×25	4,4
500	642	25×25	25×25	6,4
750	963	31×31	39×25	9,6
1000	1285	36×36	51×25	12,9
1250	1606	40×40	64×25	16,1
1500	1927	44×44	77×25	19,3
1750	2249	47×47	90×25	22,5
2000	2570	51×51	103×25	25,7

Forma. Normalmente i pilastri sono rettangolari o quadrati (fig. A, B). In casi particolari sono a L, a C, a T. Lo spessore dei pilastri che stanno lungo il contorno dell'edificio dipende dallo spessore della muratura. Di solito, però, non si scende sotto i 20 cm. Le armature longitudinali vengono disposte negli angoli e, se occorre, lungo i lati della sezione ogni 30 cm circa. Il diametro delle armature longitudinali va da 12 mm a 20 mm e solo eccezionalmente fino a 26 mm. Le staffe seguono il contorno della sezione e sono disposte a un intervallo di circa 15 cm. Il diametro delle staffe va da 6 mm a 10 mm. Il copriferro sulle staffe deve essere non meno di 2 cm (v. Acciaio, pag. 33).

Dettagli costruttivi. Riprese di armatura da piano a piano (fig. C). Pilastri di sezione particolare (fig. D). Smussi negli spigoli: è opportuno prevederli di 2,5×2,5 cm. Negli edifici in zona sismica le armature dei pilastri e quelle delle travi devono essere opportunamente collegate.

Casseri e getti. Modalità di disposizione dei casseri (fig. E). Modalità di getto: normale ma con particolare cura nella vibrazione. Tempo di maturazione: circa una settimana per tutti i pilastri di un piano.

Quantità e costi. Incidenza delle armature: da 120 a 140 kg di acciaio per m^3 di calcestruzzo. Pilastro di 30 cm×40 cm, per piano di altezza 320 cm: calcestruzzo 0,38 m^3, acciaio 50 kg, casseri 4,5 m^2. Costo (2003): 150 €.

Renato Villa

FIGURA 7: Versione elettronica della seconda edizione: la schermata della stessa scheda della figura 6

menti in LaTeX e l'organizzazione stessa del lavoro redazionale.

AG aveva fatto esperienza con i circa centocinquanta autori del Manuale Colombo per cui non ha avuto difficoltà a trovare i circa quaranta autori del Prontuario. La difficoltà è stata quella di definire assieme quali argomenti affrontare e in che modo presentarli nella forma a schede. Quindi si è data agli autori l'indicazione approssimativa di quanti caratteri dovesse contenere una pagina di testo. La redazione è poi intervenuta proponendo all'autore i tagli necessari per rientrare nello spazio prefissato. Oppure chiedendo un'integrazione del testo se era troppo corto.

Coordinare alcune decine di autori è un'impresa interessante, perché gli autori sono molto diversi fra loro sia sotto il profilo scientifico, sia sotto quello umano. Si va dal tipo pedante e preciso a quello fantasioso e disordinato. Tutti però sono interessati a vedere la propria esperienza per così dire oggettivata, nero su bianco. E appassiona provare ad estrarre in sintesi gli elementi più importanti del proprio bagaglio culturale. Molti di loro erano professionisti con poca esperienza come scrittori. Per questo l'impegno si è rivelato spesso più gravoso del previsto, ma alla fine tutti erano molto contenti del risultato.

Attualmente (luglio 2007) si sta lavorando alla terza edizione che dovrebbe uscire verso il maggio 2008. All'inizio di quest'anno si è aperto un nuovo sito internet, `http://www.manualihoepli.it/`, che dovrebbe servire a far conoscere i manuali Hoepli e a integrarli, se necessario, con pagine di aggiornamento scaricabili gratuitamente. Su questo sito si trovano anche molte pagine estratte dal Prontuario, perché si pensa che interessino e perché si spera che siano una buona pubblicità per la Hoepli.

Tra le altre modifiche che questa terza edizione conterrà, probabilmente si userà un formato maggiore, così che forse si potrà rivedere anche la scelta dei font, in particolare il loro corpo.

10 Conclusioni

Il Prontuario è stata una magnifica occasione sia per AG sia per CB per imparare come usare al meglio uno strumento poderoso come LaTeX. L'opera ha riscosso un meritato successo editoriale e, come si è detto, è in preparazione una terza edizione, ovviamente aggiornata nuovamente e privata degli inevitabili refusi residui.

Oggi creare un'opera del genere, dal punto di vista dell'utente del programma di composizione, sarebbe molto più semplice quando si tenga conto che quasi tutte le operazioni che fu necessario 'inventare' in occasione della prima edizione, sono quasi tutte già disponibili negli innumerevoli pacchetti di estensione che sono fioriti negli ultimi 10 anni.

Oggi, forse, i pacchetti disponibili consentirebbero di realizzare le macro in modo più ordinato e, specialmente, ben documentato, Mittelbach (2006); consentirebbero di realizzare le strutture compositive in modo semplice usando i comandi (ad alto livello) per gli utenti dei pacchetti di estensione; ci sarebbe più libertà nello scegliere i formati grafici, Umeki (2002), i font (tra i pacchetti standard si citano quelli per usare i font Times estesi Ryu (2000b) e quelli per usare i font Palatino estesi Ryu (2000a)), lo stile delle intestazioni delle varie sezioni Lindgren (2005), delle testatine e dei piedini van Oostrum (2004), la disposizione delle figure anche non preassemblate un una sola grande figura che riempia una intera pagina Cochran (2005).

Tuttavia questi pacchetti, benché utilissimi, permettono anche di creare delle opere che dal punto di vista del disegno editoriale potrebbero lasciare piuttosto a desiderare.

Per fortuna in questo lavoro l'interazione fra chi programmava le macro e chi ne dava le specifiche finali ha permesso di formare una particolare sinergia fra la programmazione LaTeX e l'esperienza editoriale.

Questo tipo di collaborazione ha consentito di limitare le 'fantasie' grafiche non solo grazie all'esperienza di AG, ma anche mediante l'interazione con CB, che andava via via indicando ciò che era fattibile con LaTeX e con il motore di composizione TeX.

Grazie alla collaborazione fra i due curatori, si è così evitato di imporre al design del libro specifiche irrealizzabili con il liguaggio di TeX e con l'insieme di comandi di LaTeX: che, seppur entrambi potentissimi, sono comunque limitati alle prestazione di un programma di composizione tipografico. A questo proposito, è interessante notare, per concludere, che benché TeX e LaTeX non abbiano (ancora) raggiunto la versatilità dei programmi di impaginazione usati nell'editoria, questi ultimi non sono altrettanto efficaci nella composizione dei capoversi di testo e delle formule matematiche.

Riferimenti bibliografici

Cochran, S. D. (2005). *The Subfig package*. Distribuito normalmente con il sistema TeX è contenuto nel file `~/doc/latex/subfigure/subfig.pdf`.

Colombo, G. (2003). *Manuale dell'ingegnere*. Hoepli Editore, Milano, 84ª edizione.

Guadagni, A. (a cura di) (2003). *Prontuario dell'ingegnere*. Hoepli Editore, Milano, 2ª edizione.

Lindgren, U. A. (2005). *FncyChap*. Distribuito normalmente con il sistema TeX è contenuto nel file `~/doc/latex/fncychap/fncychap.pdf`.

MITTELBACH, F. (2006). *The `doc` and `shortvrb` packages*. Distribuito normalmente con il sistema TEX è contenuto nel file `~/doc/latex/base/doc.pdf`; si veda anche il documento `~/doc/latex/base/docstrip.pdf`.

RYU, Y. (2000a). *The PX fonts*. Distribuito normalmente con il sistema TEX è contenuto nel file `~/doc/fonts/pxfonts/pxfontsdocA4.pdf`.

— (2000b). *The TX fonts*. Distribuito normalmente con il sistema TEX è contenuto nel file `~/doc/fonts/txfonts/txfontsdocA4.pdf`.

UMEKI, H. (2002). *The `geometry` package*. Distribuito normalmente con il sistema TEX è contenuto nel file `~/doc/latex/geometry/manual.pdf`.

VAN OOSTRUM, P. (2004). *Page layout in LATEX*. Distribuito normalmente con il sistema TEX è contenuto nel file `~/doc/latex/fancyhdr/fancyhdr.pdf`.

▷ Claudio Beccari
`claudio dot beccari at alice.it`

▷ Andrea Guadagni
`andrea dot guadagni at tin.it`

Introduzione a PSTricks

Massimo Caschili *

Sommario

PSTricks è un potente sistema grafico costituito da un ampio numero di estensioni; offre molti strumenti per produrre grafici, diagrammi e in generale figure con effetti ed una resa tipografica d'alta qualità.

Abstract

PSTricks is a powerful graphic system established by a large number of extensions; it offers many tools to produce pictures, graphical representations and figures with high-quality effects and an high-quality typographical performance.

1 Introduzione

Gli utenti di LaTeX hanno spesso bisogno d'inserire nei loro documenti disegni, schemi e figure di varia complessità; l'ambiente standard picture ha molti limiti, ma nell'universo di LaTeX ci sono tanti pacchetti che offrono ambienti adatti a produrre ottima grafica. In questo articolo presenterò uno dei sistemi più completi e potenti.

PSTricks è una collezione di macro istruzioni basate su PostScript [1]. Il pacchetto di base pstricks è correlato da un insieme di altri pacchetti che ne estendono le potenzialità. Non entrerò nei dettagli del codice di PSTricks, ma è importante sapere che il sistema si basa, fra i tanti, su due file principali: `pstricks.sty` e `pstricks.tex`, pertanto l'utente non dovrà chiamare i propri file con questi nomi. In queste pagine si farà una presentazione del pacchetto principale e una veloce carrellata sui tanti pacchetti che estendono il sistema, una sorta d'indice commentato. Ogni pacchetto è rilasciato con la relativa documentazione che sarà il naturale sicuro riferimento dell'utente.

2 Il primo passo

Nella figura 1 è riportato il codice minimale per creare un semplice documento, la seconda riga richiama il pacchetto principale, nella quarta riga il comando `\red` rende rosso tutto quanto è all'interno delle parentesi graffe. I pacchetti d'estensione di PSTricks sono un buon numero, e ciascuno

```
1  \documentclass{report}
2  \usepackage{pstricks}
3  \begin{document}
4    {\red Il primo esempio}
5  \end{document}
```

Il primo esempio

FIGURA 1: Il codice essenziale per il primo esempio.

deve essere, all'occorrenza, richiamato inserendo nel preambolo (cioè prima della riga 3) il comando `\usepackage{nomepacchetto}`. Attenzione: il pacchetto `pstricks-add` deve essere caricato per ultimo. Il codice deve essere compilato con il programma **latex**. Non può essere usato il programma **pdflatex** per produrre direttamente file in formato PDF; è comunque facile ottenere un file PDF seguendo lo schema di figura 2. Il programma **latex** elabora il file di testo `file.tex` e crea un file DVI, **dvips** converte il file DVI in formato PS e il programma **ps2pdf** converte il file PS in formato PDF.

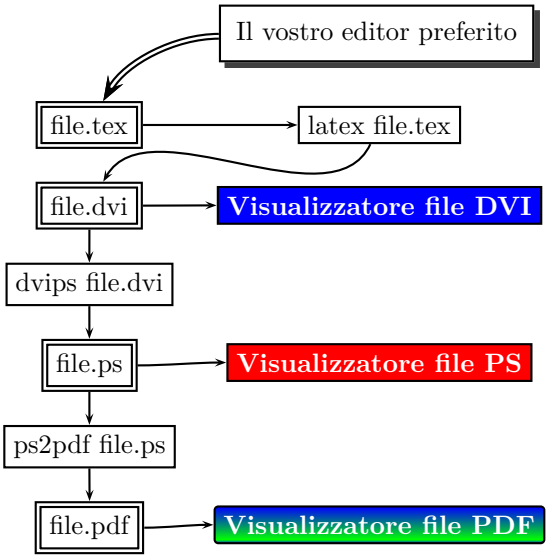


FIGURA 2: Questa figura è essa stessa un esempio delle possibilità offerte da PSTricks, il codice è nella sezione dedicata a `pst-node`.

I visualizzatori del formato `DVI` possono dare qualche problema con PSTricks, non si tratta di errori rilevati in fase di compilazione, ma di visualizzazione errata. Convertendo il file `*.dvi` in formato PostScript il risultato è perfetto. L'ulteriore conversione in formato `PDF` è ugualmente perfetta.

Chi usa la linea di comando, deve dare uno die-

*Desidero ringraziare: Enrico Bini per aver ben presentato questo lavoro in mia vece, il revisore per la sua pazienza e i suoi preziosi consigli.

1. PostScript è un linguaggio di descrizione di pagina interpretato, sviluppato da Adobe Systems. Non è necessario conoscere il PostScript per usare PSTricks.

tro l'altro i seguenti comandi oppure inserirli in un batch:

```
latex file.tex
dvips -D800 file.dvi
ps2pdf -sPAPERSIZE#a4 file.ps file.pdf
```

3 Le basi di pstricks

In questa sezione illustrerò le principali caratteristiche del pacchetto, i comandi di base, i parametri. I comandi di PSTricks hanno il loro argomento fra parentesi graffe, gli argomenti opzionali fra parentesi quadre. Il segno di uguale associa un valore ad un parametro valido per quel comando. Le parentesi tonde racchiudono i valori per le coordinate. PSTricks fornisce un ambiente analogo a picture che si chiama pspicture. La figura 3 illustra la sintassi ed evidenzia l'area delimitata dall'ambiente, l'uso delle coordinate permette d'inserire gli oggetti ovunque nell'area e anche fuori di essa. Gli oggetti con coordinate esterne all'area *protetta* potranno andare a sovrapporsi a qualunque elemento della pagina.[2] La variante con asterisco pspicture* evita che un oggetto oltrepassi i limiti dell'area dell'ambiente, in pratica gli oggetti saranno tagliati in corrispondenza del perimetro dell'area delimitata dall'ambiente.

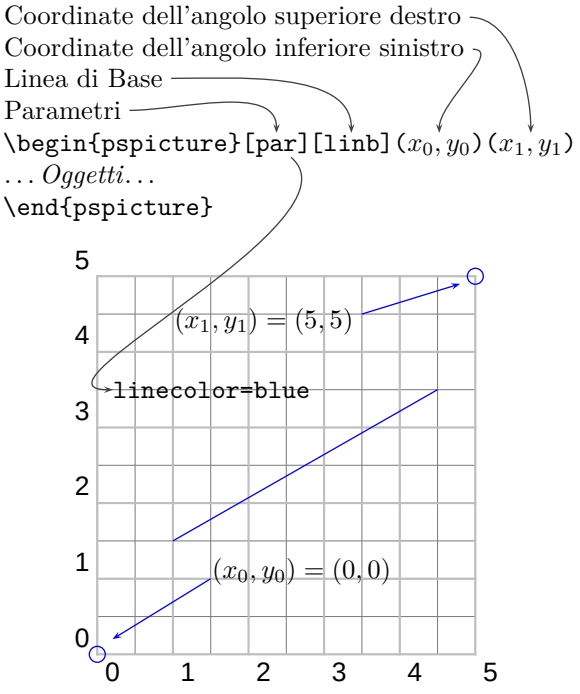


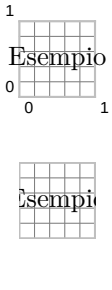
Figura 3: Sintassi ed esempio dell'ambiente pspicture. Si noti che l'indicazione del parametro linecolor=blue ha effetto per tutti gli oggetti che fanno uso di quel parametro. I parametri inseriti nella dichiarazione dell'ambiente hanno valore solo all'interno dell'ambiente.

La variante con asterisco, della sintassi illustrata nella figura 3, è:

2. Questi concetti di base sono comuni all'ambiente standard picture. Vedi bibliografia.

```
\begin{pspicture*}[par][linb](x0, y0)(x1, y1)
... Oggetti...
\end{pspicture*}
```

Nella figura 4 è illustrata la differenza di comportamento fra l'ambiente pspicture con e senza l'asterisco.




```
\begin{pspicture}(0,0)(1,1)
\psgrid[gridcolor=lightgray,
        gridlabels=6pt](0,0)(1,1)
\rput[c](0.5,0.5){Esempio}
\end{pspicture}
```



```
\begin{pspicture*}(0,0)(1,1)
\psgrid[gridcolor=lightgray,
        gridlabels=6pt](0,0)(1,1)
\rput[c](0.5,0.5){Esempio}
\end{pspicture*}
```

Figura 4: L'ambiente pspicture e pspicture*.

Per fissare la posizione degli oggetti ci sono diversi comandi, che possono essere usati anche fuori dall'ambiente pspicture. Ci sono due varianti, una con l'asterisco e l'altra senza. I comandi d'inserimento sono i seguenti:

$$\texttt{\textbackslash rput*[rif]\{angolo\}}(x_0, y_0)\texttt{\{ogg.\}}$$

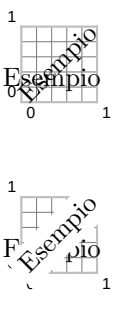
Qualora un oggetto, inserito con la variante asterisco, si sovrapponga ad un altro oggetto, questo non si vedrà più in trasparenza, ma sarà completamente coperto, si veda l'esempio 5.

$$\texttt{\textbackslash uput*\{dist\}[rif]\{angolo\}}(x_0, y_0)\texttt{\{ogg.\}}$$

Questo comando permette d'inserire un oggetto, alle coordinate indicate, in modo particolare: l'oggetto è inserito, rispetto al suo punto di riferimento rif, a una distanza dist dalle coordinate (x_0, y_0), in direzione indicata da angolo.

$$\texttt{\textbackslash cput*[par]\{angolo\}}(x_0, y_0)\texttt{\{ogg.\}}$$

Questo comando crea un cerchio, nello stile indicato in par, attorno all'oggetto in (x_0, y_0), che può essere ruotato di un certo angolo.




```
\begin{pspicture}(0,0)(1,1)
\psgrid[gridcolor=lightgray,
        gridlabels=6pt](0,0)(1,1)
\rput(0.4,0.3){Esempio}
\rput{45}(0.5,0.5){Esempio}
\end{pspicture}
```



```
\begin{pspicture}(0,0)(1,1)
\psgrid[gridcolor=lightgray,
        gridlabels=6pt](0,0)(1,1)
\rput(0.4,0.3){Esempio}
\rput*{45}(0.5,0.5){Esempio}
\end{pspicture}
```

Figura 5: Si noti che la variante con asterisco si sovrappone, coprendo, l'oggetto sottostante.

Un qualunque oggetto può essere considerato come se fosse inscritto in un rettangolo. Il punto di

riferimento predefinito è l'incrocio delle diagonali, detto impropriamente centro dell'oggetto. Quando s'inserisce un oggetto con i comandi di posizione, il punto di riferimento coincide con le coordinate indicate con il comando. E' possibile modificare il punto di riferimento dando al parametro `rif` i valori indicati in figura 6.

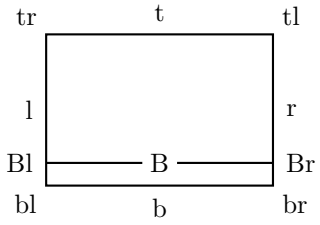


FIGURA 6: I valori del parametro `rif` sono riportati in prossimità del punto di riferimento che individuano, si noti la linea di base.

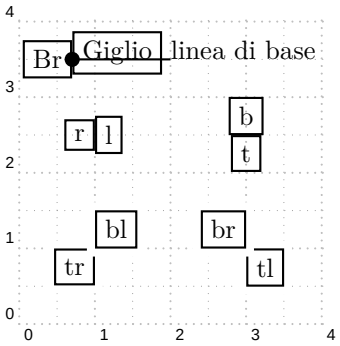


FIGURA 7: Applicazione dei parametri di posizione. In ogni rettangolo è indicato il punto di riferimento usato per l'inserimento dell'oggetto. L'esempio in alto evidenzia la linea di base e il punto di riferimento della parola nel *suo* rettangolo.

Con il comando `\multirput` è possibile inserire oggetti un numero di volte prefissato. La sintassi di questo comando è:

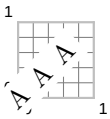
`\multirput*[rif]{`α`}(`x_0, y_0`)(`$\Delta x, \Delta y$`){N}{O}`

Questo comando è una variante di `\rput`, il parametro `rif` ha il significato già visto. Gli oggetti `O` sono inseriti a partire dalle coordinate cartesiane (x_0, y_0) un numero di volte pari al numero intero `N`. Gli oggetti sono posti, distanziati, orizzontalmente e verticalmente dalle distanze $(\Delta x, \Delta y)$, possono essere anche ruotati di un angolo α. Esiste un altro comando più efficiente per oggetti grafici più complessi:

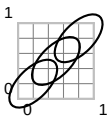
`\multips{`α`}(`x_0, y_0`)(`$\Delta x, \Delta y$`){N}{OGraf}`

3.1 Parametri

Gli elementi grafici in PSTricks hanno, ciascuno, proprie caratteristiche modificabili attraverso l'uso di parametri. Ogni comando può essere seguito da un gruppo, delimitato da parentesi quadre, che



```
\begin{pspicture}(0,0)(1,1)
   \psgrid[gridcolor=lightgray,
        gridlabels=6pt](0,0)(1,1)
\multirput*{45}(0,0)(.3,.3){3}{A}
\end{pspicture}
```



```
\begin{pspicture}(0,0)(1,1)
\psgrid[gridcolor=lightgray,
        gridlabels=6pt](0,0)(1,1)
\multips{45}(.2,.2)(.3,.3){3}{%
          \psellipse(0,0)(.4,.2)}
\end{pspicture}
```

FIGURA 8: Applicazione dei comandi `\multips` e `\multirput*`.

contiene i parametri, con relativo valore, separati da una virgola. Per cambiare i valori dei parametri predefiniti, globalmente, si usa il seguente comando con la sintassi:

`\psset{par1=val1, ..., parN=valN}`

Se il comando `\psset` è all'interno di un ambiente, i parametri assumono i valori indicati, per ogni oggetto grafico inserito nel medesimo ambiente; al di fuori dell'ambiente i valori attivi sono quelli predefiniti. Se il comando è posto fuori da qualunque ambiente, il valore dei parametri influenza tutti gli oggetti che seguono, a parte quelli protetti da un particolare ambiente.

In sintesi, con la sintassi:

```
\comando[par1=val1,.., parN=valN](..){..}
```

I parametri hanno effetto solo per il comando specificato.

Con la sintassi:

```
\begin{ambiente}[par1=val1,...,parN=valN]

...

\end{ambiente}
```

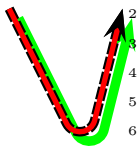
I parametri hanno effetto solo localmente, l'esempio seguente è equivalente al precedente.

```
\begin{ambiente}
 \psset{par1=val1,..., parN=valN}

 ...

\end{ambiente}
```

Il parametri sono numerosi, alcuni agiscono su più oggetti, altri hanno parametri peculiari. Di seguito è riportata una lista di alcuni dei parametri usati in questo articolo. La lettura della documentazione ufficiale di ciascun pacchetto è in questo caso consigliabile. Poiché è possibile indicare più parametri, separati da una virgola, il loro effetto è cumulativo. Come s'intuisce le combinazioni sono innumerevoli. Il valore dei parametri è assegnato con il segno di uguale.



```
\begin{pspicture}(0,0)(1.8,2)
\psline[linestyle=dashed,
 dash=5pt 1pt,doubleline=true,
 doublecolor=red,linearc=.2,
 doublesep=2pt,shadow=true,
 shadowsize=5pt,shadowangle=-45,
 shadowcolor=green]
 {->}(0,2)(1,0)(1.5,2)
 \end{pspicture}
```

FIGURA 9: Esempio dell'uso cumulativo dei parametri.

linewidth Controlla lo spessore del tratto del disegno. Per una linea sarà lo spessore della linea, per un cerchio o un poligono sarà lo spessore del tratto del perimetro. Valore predefinito `0.8pt`.

linecolor Controlla il colore delle linee e curve il cui tratto ha lo spessore indicato da `linewidth`. Valore predefinito `black`.

fillstyle Stile di riempimento delle figure. Se `fillstyle=solid` il colore uniforme di riempimento sarà dato da `fillcolor`. Valore predefinito `none`.

fillcolor Colore di riempimento.

showpoints Mostra i punti di riferimento.

linearc Controlla l'arrotondamento degli spigoli in una spezzata o un poligono. Il valore indicato è il raggio di curvatura dell'arco. Valore predefinito `0pt`.

framearc Come il precedente ma per `\psframe`. I valori possibili sono fra 0 e 1. Valore predefinito `0`.

linestyle Determina come una linea, o curva, debba essere tracciata. I valori possibili sono `none`, `solid`, `dashed`, `dotted`. Valore predefinito `solid`.

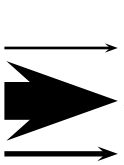
dash Il tratteggio è definito dalla lunghezza dei segmenti `d1` del tratteggio e dalla loro spaziatura `d2`. Valore predefinito `5pt 3pt`.

3.2 Dimensioni

L'unità di misura predefinita è il centimetro, è comunque possibile modificare l'unità assegnando alla grandezza direttamente il valore. Ad esempio l'oggetto linea ha come caratteristica il proprio spessore modificabile con il parametro `linewidth`.

PSTricks fa uso di registri per fissare i valori dei parametri dimensionali. Questi valori possono essere modificati con i comandi:

```
\pssetlength{reg}{dim}
\psaddtolength{reg}{dim}
```



```
\begin{pspicture}(1.7,2)
\rput(0,1.7){%
 \psline[linewidth=1pt]{->}(1.5,0)}
\rput(0,1){%
 \psline[linewidth=.5]{->}(1.5,0)}
\psset{linewidth=2pt}
\rput(0,0.3){\psline{->}(1.5,0)}
\end{pspicture}
```

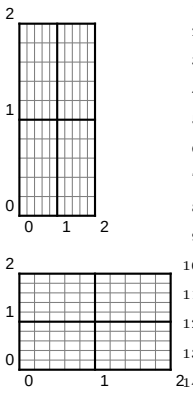
FIGURA 10: Nel primo esempio lo spessore del segmento è esplicitato con l'unità di misura; nel secondo esempio manca l'unità: è attiva quella predefinita (il centimetro). Di seguito il comando `\psset` fissa una nuova unità che influenza il seguente oggetto.

Registro	Parametro
\psunit	unit
\psxunit	xunit
\psyunit	yunit
\psrunit	runit

TABELLA 1: Registri e parametri: il valore di ciascun parametro è associato al registro corrispondente, il primo è usato per lunghezze generiche, il secondo e terzo per le dimensioni in un sistema di coordinate cartesiane, l'ultimo per coordinate polari. I valori predefiniti sono per tutti uguali ad 1 cm.

L'argomento `reg` è uno dei registri riportati nella tabella 1. L'uso di questi comandi è analogo a quelli standard di LaTeX `\setlength` e `\addtolength`.

Le coordinate possono essere regolate con opportuni cambiamenti di unità, nella figura 11 è illustrato l'uso dei parametri `xunit` e `yunit`.




```

\begin{pspicture}(0,0)(2,3)
\psgrid[xunit=0.5cm,
        yunit=.5in,
        gridlabels=6pt]
        (0,0)(2,2)
\end{pspicture}

\begin{pspicture}(0,0)(2,2)
\psgrid[xunit=1cm,
        yunit=0.25in,
        gridlabels=6pt]
        (0,0)(2,2)
\end{pspicture}
```

FIGURA 11: Cambiamento di unità per le coordinate: i valori numerici sono gli stessi, ma con unità di misura differenti.

Con il comando `\degrees[N]` si può cambiare l'unità di misura per gli angoli. Il parametro `N` indica il numero di unità per ciclo (360 gradi), senza argomenti equivale a `\degrees[360]`. Il comando `\radians` equivale a `\degrees[6.28319]`

3.3 Linee e Poligoni

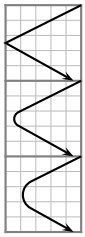
Una linea può essere generata con l'uso del comando `\psline`, ecco la sintassi generale:

`\psline*[par]{term}`$(x_0, y_0) \ldots (x_n, y_n)$

Le coordinate indicano gli estremi del segmento che si vuole tracciare. Indicando più di due punti, il comando disegna una spezzata. Con il parametro `linearc` si possono arrotondare gli angoli della spezzata, il valore indicato è quello del raggio dell'arco che raccorda i segmenti.

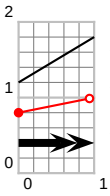
Il comando `\qline`$(x_1, y_1)(x_2, y_2)$ senza parametri traccia un semplice segmento, il comportamento è del tutto analogo a `\psline` passante per due punti.

L'argomento `term` fra parentesi graffe specifica il tipo di terminatori che possono essere usati, agli estremi della linea, nella figura 16 sono indicati gli stili e il comando per generarli.



```
\psline{->}(1,1)(0,.5)(.9,0)
\psline[linearc=.1]{->}(1,1)(0,.5)(.9,0)
\psline[linearc=.2]{->}(1,1)(0,.5)(.9,0)
```

FIGURA 12: Questi tre esempi sono identici a parte il valore del parametro `linearc`. Si noti che al crescere del parametro, l'arco di raggio maggiore allontana la linea dal punto `(0,.5)`. Vedi anche la figura 15.



```
\psline(0,1.2)(1,1.8)
\psline[linecolor=red]{*-oo}(0,.8)(1,1)
\psline[linewidth=3pt]{->>}(0,.4)(1,.4)
```

FIGURA 13: Esempi d'uso del comando `\psline`. Si noti l'uso dei parametri fra parentesi quadre.

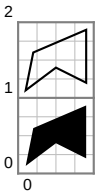
Per tracciare poligoni si usa il comando:

`\pspolygon*[par]`$(x_0, y_0)(x_1, y_1) \ldots (x_n, y_n)$

Questo comando genera una spezzata chiusa fra le ultime coordinate indicate e le prime. Se si omette l'asterisco sarà tracciato il perimetro, con l'asterisco tutta l'area del poligono sarà riempita. Gli angoli possono essere arrotondati specificando il parametro `linearc`.

Per disegnare cornici sono disponibili diversi comandi con particolari caratteristiche. Una semplice cornice si ottiene con il comando:

`\psframe*[par]`$(x_0, y_0)(x_1, y_1)$



```
\pspolygon(1,1)(5,4)(9,2)(9,9)(2,6)
\pspolygon*(1,1)(5,4)(9,2)(9,9)(2,6)
```

FIGURA 14: Poligoni con il comando `\pspolygon` e `\pspolygon*`.

```
\begin{pspicture}[xunit=1cm,yunit=1cm]
                 [baseline=0.1](0,0)(7,2)
  \psset{gridcolor=gray,gridlabels=6pt}
 \psgrid(0,0)(0,0)(7,2)
 \psset{linecolor=red,linewidth=.5pt,
        linestyle=dashed}
 \psline(0,0)(7,2)
 \psline(1,1.5)(7,2)
 \psline(0,1.5)(.5,.5)
 \psline(1,1.5)(.5,.5)
 \psset{linecolor=blue,linestyle=solid,
        linewidth=.8pt}
 \psline[linearc=3mm]{->}
        (0,1.5)(.5,.5)(1,1.5)(7,2)(0,0)
\end{pspicture}
```

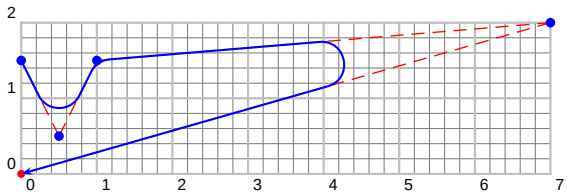


FIGURA 15: In questo esempio si vede come funziona il parametro `linearc`. Si notino le tangenti alla curva, ed i punti di riferimento.

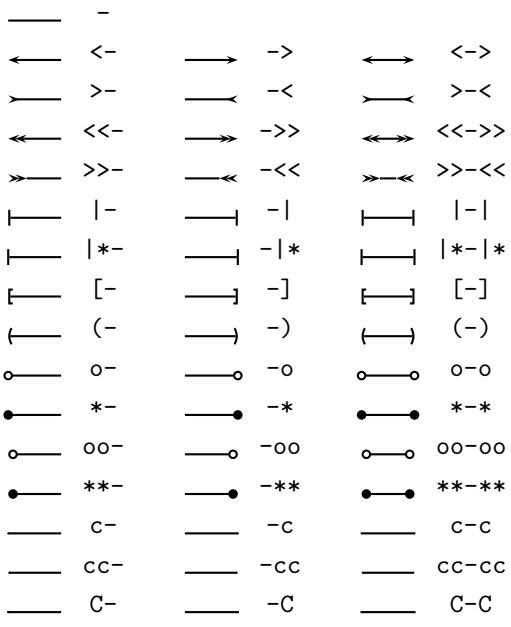


FIGURA 16: In questi tre gruppi di colonne sono riportati gli stili dei terminatori di linea utilizzabili. Il trattino indica la linea (segmento o curva). I simboli che creano i terminatori possono essere anche solo su un estremo o combinati fra loro.

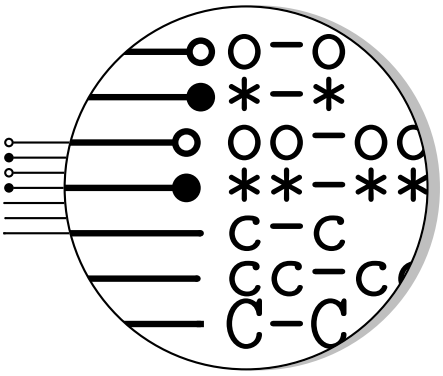


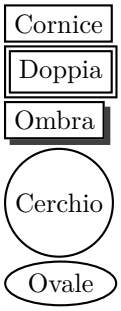
FIGURA 17: Alcuni estremi *sembrano* uguali, occhio ai dettagli.

Le coordinate da indicare sono quelle dell'angolo inferiore sinistro (non obbligatorio) e quelle dell'angolo superiore destro. Senza l'asterisco, il comando disegna una cornice rettangolare, con l'asterisco riempie l'area della figura.

I seguenti comandi generano varianti di cornici:

```
\psframebox*[par]{Ogg}
\psdblframebox*[par]{Ogg}
\psshadowbox*[par]{Ogg}
\pscirclebox*[par]{Ogg}
\psovalbox*[par]{Ogg}
```

Nella figura 18 sono riportati alcuni esempi senza parametri.




```
1 \psframebox{Cornice}
2
3 \psdblframebox{Doppia}
4
5 \psshadowbox{Ombra}
6
7 \pscirclebox{Cerchio}
8
9 \psovalbox{Ovale}
```

FIGURA 18: Esempi di cornici.

Per illustrare la varietà degli effetti dei parametri nella figura 19 gli esempi sono stati costruiti sul comando `\psframebox`.

Per costruire cornici con ombreggiatura si usa il comando `\psshadowbox`, in figura 20 è illustrato un esempio.

L'uso di questi comandi mischia grafica e testo con un'ottima resa.

Tutto ciò è ottenuto con il codice seguente:

```
\psframebox{L'uso di questi comandi} mischia
\psdblframebox[linecolor=red,linestyle=dotted,
linewidth=2pt]{grafica} e \psdiabox{testo} con
un'\psframebox[fillstyle=solid,fillcolor=gray]
{\bfseries\color{white}ottima resa}.
```

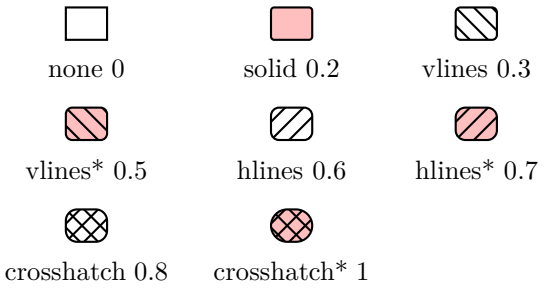


FIGURA 19: Ciascun elemento è ottenuto con il comando `\psframebox`: è indicato il valore di `fillstyle` e il valore di `framearc`, gli altri parametri rimangono gli stessi; `fillcolor=pink`. Si noti il comportamento del parametro con asterisco.

```
1 \psshadowbox[fillstyle=solid,
2              fillcolor=yellow]
3  {\color{red}
4   \begin{tabular}{c}
5     Una scatola\\colore ed ombra
6   \end{tabular}}
```

Una scatola
colore ed ombra

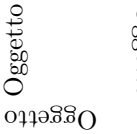
FIGURA 20: Il colore `yellow` è fra quelli predefiniti, nella sezione 3.8 è spiegato come ottenere altri colori personalizzati.

Per ottenere effetti di riempimento graduale si può attivare il parametro `fillstyle=gradient`. In questo caso è necessario caricare il pacchetto pst-grad. Si veda la sezione dedicata.

Per ruotare degli oggetti si usa il comando

```
\rotateleft{Oggetto}
\rotateright{Oggetto}
\rotatedown{Oggetto}
```

Nella figura 21 sono visibili tre semplici esempi.




```
1 \rotateleft{Oggetto}
2 \rotateright{Oggetto}
3 \rotatedown{Oggetto}
```

FIGURA 21: Ribaltamenti.

3.4 Archi, cerchi ed ellissi

Per disegnare un cerchio (o disco) si può usare il comando senza argomenti:

```
\qdisk(xc, yc){raggio}
```

Le coordinate da indicare sono quelle del centro, l'argomento è la misura del raggio. Il comando che offre completa flessibilità è:

```
\pscircle*[par](xc, yc){raggio}
```

Anche qui le coordinate sono quelle del centro, con l'asterisco si ottiene un cerchio, senza asterisco una circonferenza, con i parametri è possibile modificare caratteristiche quali il colore. Se si vuole disegnare solo un settore circolare il comando è:

`\pswedge*[par]`(x_c, y_c)`{raggio}{ang1}{ang2}`

Qui oltre alle coordinate del centro e alla misura del raggio, occorre indicare l'angolo iniziale del settore e quello finale.

Per disegnare un semplice arco si usa il comando:

`\psarc*[par]{term}`(x_c, y_c)`{raggio}{ang1}{ang2}`

La variante senza asterisco traccia un semplice arco, con l'asterisco si genera una figura *piena* fra l'arco e la corda della circonferenza fra gli estremi dell'arco. Per il centro, il raggio e gli angoli vale quanto detto per il comando `\pswedge`, per l'argomento `ter` vale quanto detto per il comando `\psline`, i valori possibili sono riportati in figura 16. Per disegnare ellissi il comando è:

`\psellipse*[par]`(x_c, y_c)`(AsseOr,AsseVert)`

Anche con questo comando c'è la variante senza asterisco e con asterisco (figura piena). Occorre indicare le coordinate del centro, la misura del semiasse maggiore (orizzontale) e quella del semiasse minore (verticale).

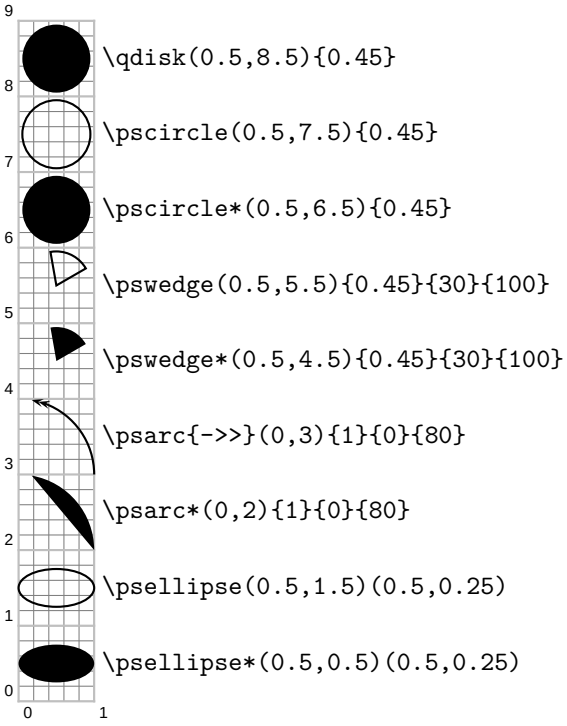


FIGURA 22: Cerchi, ellissi, settori e archi.

3.5 Curve

Per costruire e tracciare semplici curve è disponibile il comando `\psbezier`. Il comando è seguito dalle coordinate di quattro punti. I primi due (x_i, y_i) e (x_{t1}, y_{t1}) individuano la tangente di un estremo della curva (x_i, y_i). Gli altri due punti (x_{t2}, y_{t2}) e (x_f, y_f), individuano la tangente dell'altro estremo (x_f, y_f).

`\psbezier*[par]{term}`
$(x_i, y_i)(x_{t1}, y_{t1})(x_{t2}, y_{t2})(x_f, y_f)$

I parametri `par` sono gli stessi validi per `\psline`, lo stesso vale per i terminatori `term`.

```
\psbezier[showpoints=true](0,2)(.5,.5)(6,2)(7,0)
```

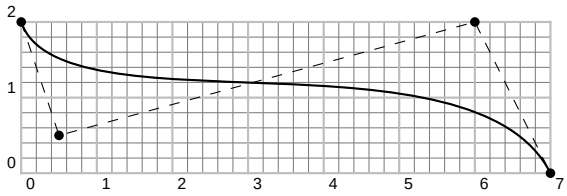


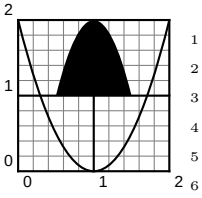
FIGURA 23: Esempio di curva `\psbezier`.

Per tracciare una parabola esiste un comando specifico:

`\parabola*[par]{term}`$(x_i, y_i)(x_{Mm}, y_{Mm})$

La parabola è disegnata a partire dalla coordinata (x_i, y_i) fino al suo punto di massimo (o minimo) indicato con (x_{Mm}, y_{Mm}). Il comando completa la parabola per simmetria.

La variante con asterisco *riempie* l'area concava dalla parabola fino alla congiungente degli estremi dell'arco. I valori dell'argomento `term` sono indicati in figura 16.



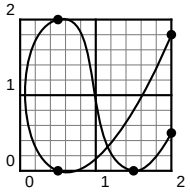
```
\begin{pspicture}(2,2)
 \psgrid[gridlabels=6pt]
        (0,0)(2,2)
 \parabola(0,2)(1,0)
 \parabola*(.5,1)(1,2)
\end{pspicture}
```

FIGURA 24: Esempio di parabola.

Ci sono altri tre comandi per tracciare curve (vedi le figure 25-29) la cui sintassi è la seguente:

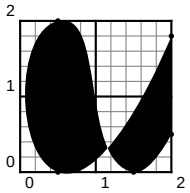
`\pscurve*[par]{sti}`$(x_1, y_1) \ldots (x_n, y_n)$
`\psccurve*[par]{sti}`$(x_1, y_1) \ldots (x_n, y_n)$
`\psecurve*[par]{sti}`$(x_1, y_1) \ldots (x_n, y_n)$

Questi tre comandi disegnano la curva per interpolazione fra i punti indicati. L'argomento `sti` individua lo stile degli estremi della curva in modo analogo con quanto visto per il comando `\psline` secondo la figura 16.



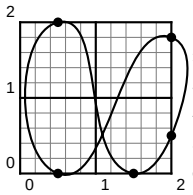
```
\begin{pspicture}
    [showpoints=true](2,2)
 \psgrid[gridlabels=6pt]
                    (0,0)(2,2)
 \pscurve(2,1.8)(.5,0)(.5,2)
                  (1.5,0)(2,.5)
\end{pspicture}
```

FIGURA 25: In questo esempio è stato valorizzato il parametro opzionale `showpoints` per evidenziare i punti di riferimento.



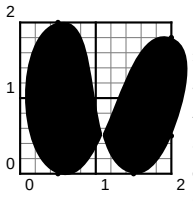
```
\begin{pspicture}
    [showpoints=true](2,2)
 \psgrid[gridlabels=6pt]
                    (0,0)(2,2)
 \pscurve*(2,1.8)(.5,0)(.5,2)
                  (1.5,0)(2,.5)
\end{pspicture}
```

FIGURA 26: Come per la figura 25 ma con la variante asterisco.



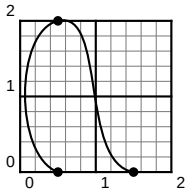
```
\begin{pspicture}
    [showpoints=true](2,2)
 \psgrid[gridlabels=6pt]
                    (0,0)(2,2)
 \psccurve(2,1.8)(.5,0)(.5,2)
                  (1.5,0)(2,.5)
\end{pspicture}
```

FIGURA 27: In questo esempio il comando `\psccurve` chiude la curva tra il primo e l'ultimo punto indicato.



```
\begin{pspicture}
    [showpoints=true](2,2)
 \psgrid[gridlabels=6pt]
                    (0,0)(2,2)
 \psccurve*(2,1.8)(.5,0)(.5,2)
                  (1.5,0)(2,.5)
\end{pspicture}
```

FIGURA 28: Variante con asterisco dell'esempio di figura 27.



```
\begin{pspicture}
    [showpoints=true](2,2)
 \psgrid[gridlabels=6pt]
                    (0,0)(2,2)
 \psecurve(2,1.8)(.5,0)(.5,2)
           (1.5,0)(2,.5)
\end{pspicture}
```

FIGURA 29: L'andamento della curva tiene conto di tutti i punti, ma omette di tracciare le parti che collegano il primo e l'ultimo punto.

3.6 Punti

PSTricks offre una varietà di stili di *punti* il cui uso è limitato solo dalla fantasia e dal gusto. Nella tabella 2 sono presentati gli stili disponibili e nella figura 30 alcuni esempi.

La sintassi dell'oggetto *punto* è la seguente:

`\psdots*[par]`$(x_1, y_1)(x_2, y_2)\ldots(x_n, y_n)$

I parametri possibili sono:

`dotstile=stile` Il valore di `stile` è indicato nella tabella 2.

`dotscale=num1 num2` Deforma il punto di tipo `dotstile=stile` con scala `num1` orizzontalmente e con scala `num2` verticalmente.

`dotangle=angolo` Ruota il simbolo scelto dell'angolo indicato.

Stile	Esempio
Predefinito	● ● ● ●
o	○ ○ ○ ○
+	+ + + +
x	× × × ×
asterisk	* * * *
diamond	◊ ◊ ◊ ◊
diamond*	♦ ♦ ♦ ♦
oplus	⊕ ⊕ ⊕ ⊕
otimes	⊗ ⊗ ⊗ ⊗
triangle	△ △ △ △
triangle*	▲ ▲ ▲ ▲
square	□ □ □ □
square*	■ ■ ■ ■
pentagon	⬠ ⬠ ⬠ ⬠
pentagon*	⬟ ⬟ ⬟ ⬟
|	| | | |

TABELLA 2: Varietà di stili per i punti

```
\begin{pspicture}(1,2)
 \psgrid[gridlabels=6pt,
         subgridcolor=lightgray](0,0)(1,2)
 \psdots*[dotstyle=triangle*](0.4,.5)(0.7,.5)
 \psdots*[dotstyle=triangle*,dotscale=2 3,
          dotangle=45](0.3,1.5)(0.6,1.5)
\end{pspicture}
```

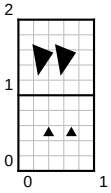


FIGURA 30: Alcuni esempi dell'uso dell'oggetto `\psdots`.

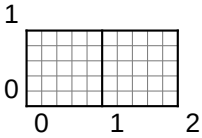
3.7 Griglie

Per chi avesse bisogno di tracciare una griglia di riferimento, PSTricks offre il comando:

`\psgrid`$(x_0, y_0)(x_{is}, y_{is})(x_{sd}, y_{sd})$

Il comando usa le coordinate di tre punti:(x_0, y_0) per indicare l'origine del sistema di coordinate, (x_{is}, y_{is}) indica la posizione dell'angolo inferiore sinistro e l'angolo superiore destro è

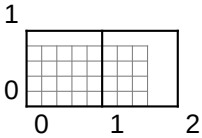
dato da (x_{sd}, y_{sd}). Se viene omessa l'origine, questa coinciderà con l'angolo inferiore sinistro. Usando il comando all'interno dell'ambiente pspicture si possono omettere le coordinate di tutti e tre i punti, in tal modo la griglia è tracciata secondo le dimensioni dell'ambiente. Il comando è correlato da parametri opzionali riepilogati nella tabella 3.




```
\begin{pspicture}(2,1)
 \psgrid
\end{pspicture}
```

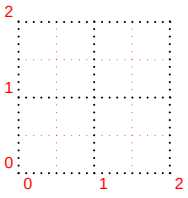
FIGURA 31: Esempio d'uso di \psgrid senza parametri e coordinate all'interno dell'ambiente pspicture

La struttura della griglia è costruita per multipli interi dell'unità **xunit** e **yunit**, è possibile cambiare l'unità di misura direttamente come mostrato in figura 32 e figura 11.



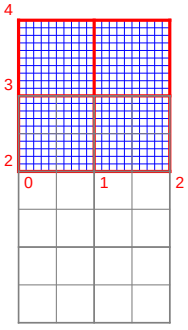

```
\begin{pspicture}(2,1)
 \psgrid(0,0)(45pt,25pt)
\end{pspicture}
```

FIGURA 32: In questo caso sono state indicate le unità di misura. La griglia principale è stata disegnata con le dimensioni dell'ambiente pspicture.




```
\begin{pspicture}(2,2)
\psgrid[griddots=10,
 gridlabels=6pt,
 gridlabelcolor=red,
 subgriddiv=2,
 subgridcolor=red,
 subgriddots=5](0,0)(2,2)
\end{pspicture}
```

FIGURA 33: In questo caso sono stati valorizzati diversi parametri, si noti la dimensione delle etichette e la diversa divisione della griglia con l'uso dei punti in luogo della linea continua.




```
\begin{pspicture}(2,4)
\psgrid[gridwidth=1.2pt,
gridcolor=red,
gridlabels=6pt,
gridlabelcolor=red,
subgriddiv=10,
subgridcolor=blue,
subgridwidth=.2pt](0,2)(2,4)
\psgrid[gridwidth=0.6pt,
gridcolor=gray,
gridlabels=0pt,
gridlabelcolor=red,
subgriddiv=2](0,0)(2,3)
\end{pspicture}
```

FIGURA 34: Due griglie sovrapposte. Sono state modificate le dimensioni dello spessore delle linee della griglia e della sottogriglia. Si noti che per eliminare le etichette della seconda griglia si è posto `gridlabels=0pt`. Altri parametri sono illustrati nella tabella 3.

3.8 I colori

PSTricks offre una serie di colori e una scala di grigi predefiniti, ma con opportuni comandi, l'utente può definire e generare tutti i colori che desidera. Nella figura 35 sono indicati i comandi per ciascun colore predefinito. L'attivazione del colore ha effetto sul testo e sugli oggetti grafici che seguono il comando che attiva il colore specifico. Tale effetto può essere confinato all'interno di un gruppo delimitato dalle parentesi graffe oppure quando è contenuto in ambienti delimitati da

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

L'influenza di un comando può essere interrotta da un altro comando che attiva un diverso colore, o schermata dagli ambienti o dalle parentesi graffe.

Questo è un esempio ottenuto con il seguente codice:

```
{\blue Questo è un \red esempio}
```

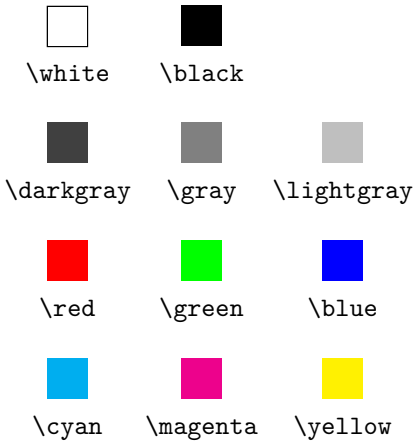


FIGURA 35: Colori predefiniti e comandi relativi

Per generare sfumature di grigio personalizzate si usa il comando:

```
\newgray{nomecolore}{num}
```

L'argomento `num` deve avere un valore compreso tra 0 (nero) e 1 (bianco).

Per richiamare il colore personalizzato si usa il comando:

```
\color{nomecolore}
```

Per generare i colori che si desiderano, occorre definirli con uno dei seguenti comandi:

```
\newrgbcolor{nomecolore}{n1 n2 n3}
 \newhsbcolor{colore}{n1 n2 n3}
\newcmykcolor{colore}{n1 n2 n3 n4}
```

Ciascuno dei precedenti comandi usa una codifica numerica. I valori numerici devono avere valori $0 \leq nx \leq 1$.

Nome	Significato	Predefinito
gridwidth	Spessore delle linee della griglia	.8pt
gridcolor	Colore delle linee della griglia	black
griddots	Punti, nel numero indicato, per ogni divisione data da subgriddiv	0
gridlabels	Dimensione dei numeri d'etichetta	10pt
gridlabelcolor	Colore per gridlabels	false
subgriddiv	Numero di linee divisorie per unità di misura	5
subgridwidth	Spessore delle linee divisorie	4pt
subgridcolor	Colore delle linee divisorie	gray
subgriddots	Come per griddots, ma per le sotto divisioni della griglia	0

TABELLA 3: Elenco dei parametri per il comando \psgrid

Esempio di grigio
con altro grigio

```
\newgray{grigiotre}{.3}
\newgray{grigiosette}{.7}
Esempio di \color{grigiotre}\ grigio\\
con \color{grigiosette}\ altro grigio
```

FIGURA 36: Tonalità personalizzate di grigio. Definizione e uso.

Per questioni di resa e compatibilità si consiglia d'usare la codifica RGB implementata dal primo comando. Le cifre della tripletta di valori definiscono il colore *mischiando* tre colori base: rosso, verde e blu.

```
\newrgbcolor{rosso}{1 0 0}
\newrgbcolor{verde}{0 1 0}
\newrgbcolor{blu}{0 0 1}
\newrgbcolor{primo}{.1 .6 0}
\newrgbcolor{secondo}{0 1 1}
\newrgbcolor{terzo}{.5 .2 .7}
{\color{rosso}\rule{15pt}{15pt}}    \\
{\color{verde}\rule{15pt}{15pt}}    \\
{\color{blu}\rule{15pt}{15pt}}      \\
{\color{primo}\rule{15pt}{15pt}}    \\
{\color{secondo}\rule{15pt}{15pt}}  \\
{\color{terzo}\rule{15pt}{15pt}}    \\
```

FIGURA 37: Colori: definizioni ed uso.

4 Le estensioni di PSTricks

Le estensioni di PSTricks sono in continua evoluzione. Il loro numero aumenta in continuazione. La seguente lista non può che essere manchevole, ma un'accurata ricerca su internet farà scoprire pacchetti per le più disparate applicazioni [3].

3. Alcuni degli esempi di questa sezione sono tratti dai manuali dei rispettivi pacchetti ai quali si riferiscono. Altri notevoli esempi si possono trovare nel sito http://www.tug.org/PSTricks/main.cgi?file=examples

pstricks

È il pacchetto di base del mondo PSTricks. La sua estensione diretta è il pacchetto pstricks-add. Per compatibilità con gli altri pacchetti correlati è importante caricare pstricks-add in coda agli altri pacchetti pst-*xxxx*.

pst-3d

Pacchetto di base per generare elementari effetti in 3D.

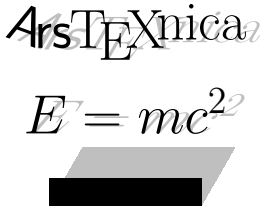


FIGURA 38: Esempio per il pacchetto pst-3d

Nella figura 38 sono mostrati gli effetti di base possibili con il pacchetto pst-3d.

Codice per la figura 38

```
\newgray{gray75}{.75}
\psset{Tshadowcolor=gray75}
\psshadow{\huge \Ars}\\[10pt]
\psshadow{\huge $E=mc^2$}\\[15pt]
\psshadow[Tshadowsize=2.5]{%
\rule{2cm}{10pt}}
```

pst-3dplot

Permette di disegnare grafici e figure in 3D.

Codice per la figura 39

```
\begin{pspicture}(-1,-1)(2,2.5)%
\pstThreeDCoor[xMin=-1,xMax=2,yMin=-1,yMax=2,
               zMin=-1,zMax=2]%
\pstThreeDBox(-1,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
\pstThreeDBox[RotX=90,linecolor=red]
             (0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
\pstThreeDBox[RotX=90,RotY=90,linecolor=green]
             (0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
```

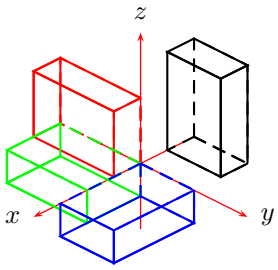


FIGURA 39: Oggetti in 3D con il pacchetto `pst-3dplot`

```
\pstThreeDBox[RotX=90,RotY=90,RotZ=90,
                          linecolor=blue]
                (0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
\end{pspicture}%
```

pstricks-add

Pacchetto addizionale che estende il pacchetto principale `pstricks` e altri due pacchetti standard: `pst-node` e `pst-plot`. È importante che `pstricks-add` sia caricato per ultimo, dopo tutti i pacchetti correlati con `pstricks`. Automaticamente è caricato `xkeyval`, `pstricks-add` dipende da questo pacchetto poiché si poggia su `pst-xkey` che è parte del pacchetto `xkeyval`.

Il pacchetto `pstricks-add` incrementa di molto la varietà di stili per gli assi cartesiani e le scale dei sistemi di riferimento. Di seguito le figure riportano alcuni esempi di scale logaritmiche e una varietà di assi riferimento.

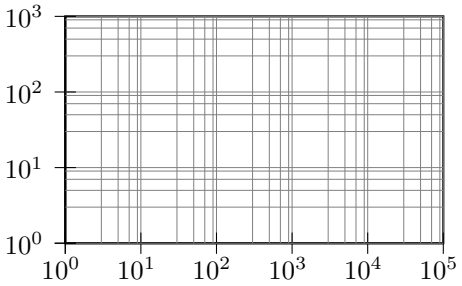


FIGURA 40: Mappe logaritmiche

Codice per la figura 40

```
\psaxes[subticks=5,axesstyle=frame,
    xylogBase=10,logLines=all](5,3)
```

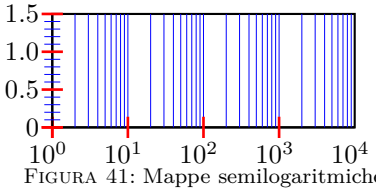


FIGURA 41: Mappe semilogaritmiche

Codice per la figura 41

```
\psaxes[axesstyle=frame,logLines=x,
  xlogBase=10,Dy=0.5,tickcolor=red,
  subtickcolor=blue,tickwidth=1pt,
  ysubticks=5,xsubticks=10](4,1.5)
```

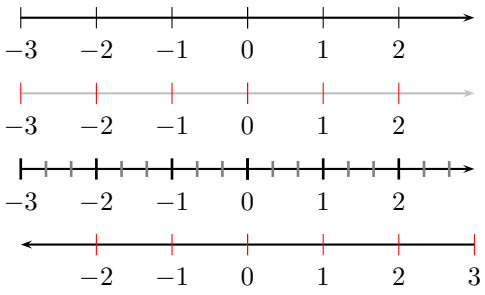


FIGURA 42: Assi

Codice per la figura 42

```
\psset{arrowscale=1}
\rput(0,3){\psaxes[yAxis=false,
  subticks=0]{->}(0,0)(-3,-3)(3,3)}
\rput(0,2){\psaxes[yAxis=false,
  subticks=0,tickcolor=red,
  linecolor=blue]{->}(0,0)(-3,-3)(3,3)}
\rput(0,1){\psaxes[yAxis=false,
  subticks=3,tickwidth=1pt,
  subtickwidth=1pt]{->}(0,0)(-3,-3)(3,3)}
\rput(0,0){\psaxes[yAxis=false,subticks=0,
      tickcolor=red]{>}(0,0)(-3,-3)(3,3)}
```

pst-abspos

Permette d'inserire oggetti tramite coordinate assolute. L'oggetto può essere inserito in qualunque punto con precisione. Per questo pacchetto l'origine delle coordinate coincide con l'angolo superiore sinistro del foglio della pagina.

pst-asr

Linguistica: permette di creare svariati diagrammi utili per questa disciplina.

pst-bar

Pacchetto per la creazione di grafici ad istogrammi. Per disegnare gli assi di riferimento occorre richiamare anche il pacchetto `pst-plot`.

pst-barcode

Permette la creazione e la stampa di codici a barre.

pst-blur

Questo pacchetto permette di ottenere ombre sfumate, meno nette di quelle prodotte con PSTricks standard. Vedi la figura 44.

pst-calendar

Genera calendari in forma tabellare e in 3D a forma di dodecaedro.


```
\begin{pspicture}(3,2.5)
 \psbarcode{12345678}
 {includetext inkspread=0.5}
 {ean8}
 \end{pspicture}

\begin{pspicture}(3,2.5)
 \psbarcode[scalex=0.7,
            scaley=0.8]
 {9781860742712}
 {includetext
  guardwhitespace}
 {ean13}
\end{pspicture}
```

FIGURA 43: Esempio per il pacchetto `pst-barcode`.

```
\begin{pspicture}(1.5,3.5)
\pscircle[shadow=true,blur=true]
          (.75,2.2){0.5}

\pscircle[shadow=true]
          (.75,0.7){0.5}
\end{pspicture}
```

FIGURA 44: Si noti la differenza: l'ombra del secondo cerchio è quella standard di PSTricks.

pst-circ

Con questo pacchetto è possibile disegnare circuiti elettronici con una buona resa tipografica. Il pacchetto offre una buona varietà di comandi per generare i componenti ideali quali bipoli, tripoli e quadripoli. Resistenze, condensatori, diodi, transistor (per citarne solo alcuni) hanno ciascuno un proprio comando. Una nutrita schiera di parametri, consente di aggiungere al circuito informazioni che completano il disegno in ogni sua parte. La sintassi è semplice ed intuitiva.

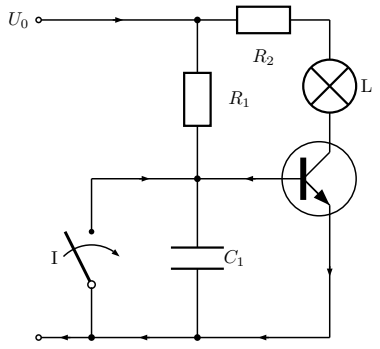


FIGURA 45: Circuiti con `pst-circ`

Codice per la figura 45

```
\begin{pspicture}(0,-1)(7,6.5)
\pnode(0,6){positivo}
\pnode(3,3){base}
\pnode([nodesep=-2] base){interruttore}
\pnode(0,0){massa}
\wire[arrows=o-*](positivo)(base|positivo)
\uput[l](positivo){$U_0$}
\resistor[labeloffset=.8](base|positivo)
                         (base){$R_1$}
\transistor[basesep=2cm](base){emettitore}
                                {colletore}
\wire[arrows=-*](interruttore)(base)
\wire(TBaseNode)(base)
\switch(interruttore|massa)(interruttore){I}
\lamp(colletore|positivo)(colletore){L}
\resistor(colletore|positivo)(base|positivo)
                                {$R_2$}
\wire(emettitore)(emettitore|massa)
\wire(emettitore|massa)(base|massa)
\capacitor(base)(base|massa){$C_1$}
\wire[arrows=*-](base|massa)(interruttore|massa)
\wire[arrows=*-o](interruttore|massa)(massa)
\end{pspicture}
```

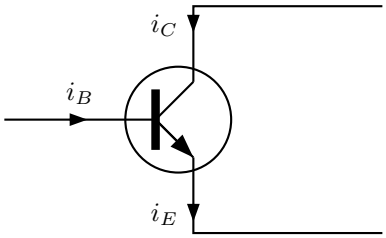


FIGURA 46: Le connessioni sono ottenute con la stessa tecnica usata per i nodi dei grafi del pacchetto `pst-node`.

Codice per la figura 46

```
\begin{pspicture}(5,3)
\pnode(0,1.5){A}
\pnode(5,0){B}
\pnode(5,3){C}
\transistor[basesep=2cm,
            transistoribaselabel=$i_B$,
       transistoricollectorlabel=$i_C$,
 transistoriemitterlabel=$i_E$](A)(B)(C)
\end{pspicture}
```

pst-coil

Permette di disegnare molle, eliche e spire.

Il codice per ottenere questi oggetti è:

```
\pscoil(0,4)(4,4)
\psset{coilwidth=0.75}
\pscoil[coilaspect=0](0,3)(4,3)
\pscoil[coilaspect=30,coilheight=0.3]
 (0,2)(4,2)
```

Le coordinate indicano gli estremi delle molle.

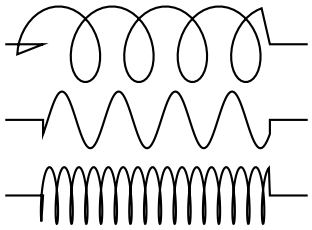


FIGURA 47: Eliche, spire e molle.

pst-dbicons

Permette di disegnare la struttura di una base dati, tabelle e loro relazioni.

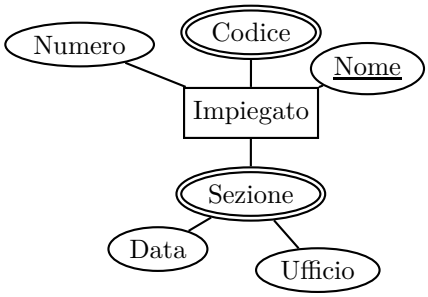


FIGURA 48: I collegamenti sono ottenuti tramite l'uso di etichette e nodi, il funzionamento ricorda il pacchetto `pst-node`.

pst-eps

Pacchetto che permette di salvare grafici da ambienti PSTricks in file in formato eps.

pst-eucl

Consente di disegnare le figure della geometria euclidea.

pst-fill

Permette di generare riempimenti con tratteggio di vario tipo, un effetto analogo a quello prodotto dai vecchi retini usati nel disegno tecnico.

pst-fr3d

Consente la costruzione di *cornici* 3D

Codice per la figura 49

```
\begin{pspicture}[doublesep=0.05](0,0)(6,1)
\multido{\nButtonA=0+0.1}{3}{%
\multido{\nButtonB=0+0.05}{3}{%
\PstFrameBoxThreeD[FrameBoxThreeDColorHSB=%
\nButtonB\space \nButtonA\space 0.8]{X1}}\\}
\end{pspicture}

\begin{psmatrix}[rowsep=1mm,colsep=1mm]
\PstFrameBoxThreeD[FrameBoxThreeDColorHSB
                   =0.6 0.3 0.5]{Premi} &
\PstFrameBoxThreeD[FrameBoxThreeDColorHSB
                   =0.4 0.1 0.5]{%
```

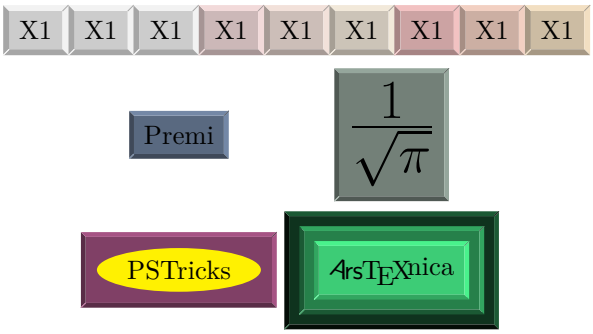


FIGURA 49: Pulsanti in 3D costruiti con `pst-fr3d`

```
\Huge $\frac{1}{\sqrt{\pi}}$ \\
\PstFrameBoxThreeD[FrameBoxThreeDColorHSB
                   =0.9 0.5 0.5]{%
\psovalbox[fillstyle=solid,fillcolor=yellow]
                   {PSTricks}}          &
\PstFrameBoxThreeD[FrameBoxThreeDColorHSB
                   =0.4 0.7 0.2]{%
\PstFrameBoxThreeD[FrameBoxThreeDColorHSB
                   =0.4 0.7 0.5]{%
\PstFrameBoxThreeD[FrameBoxThreeDColorHSB
                   =0.4 0.7 0.8]{\Ars}}}
\end{psmatrix}
```

pst-func

Per disegnare il grafico di speciali funzioni matematiche.

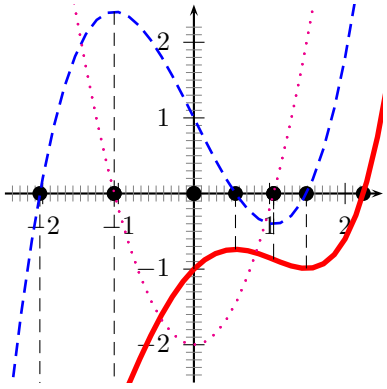


FIGURA 50: Grafico generato con `pst-func`.

Codice per la figura 50

```
\begin{pspicture*}(-2.5,-2.5)(2.5,2.5)
\psaxes{->}(0,0)(-2.5,-2.5)(2.5,2.5)%
\psset{dotscale=2}
\psPolynomial[markZeros,linecolor=red,
linewidth=2pt,coeff=-1 1 -1 0 0.15]{-4}{3}
\psPolynomial[markZeros,linecolor=blue,
linewidth=1pt,linestyle=dashed,%
coeff=-1 1 -1 0 0.15,Derivation=1,
zeroLineTo=0]{-4}{3}%
\psPolynomial[markZeros,linecolor=magenta,
linewidth=1pt,linestyle=dotted,%
```

```
coeff=-1 1 -1 0 0.15,Derivation=2,
zeroLineTo=0]{-4}{3}%
\psPolynomial[markZeros,linecolor=magenta,
linewidth=1pt,linestyle=dotted,%
coeff=-1 1 -1 0 0.15,Derivation=2,
zeroLineTo=1]{-4}{3}%
\end{pspicture*}
```

pst-geo

Consente di disegnare mappe di stati e continenti.

pst-gr3d

Questo pacchetto consente di disegnare griglie tridimensionali.

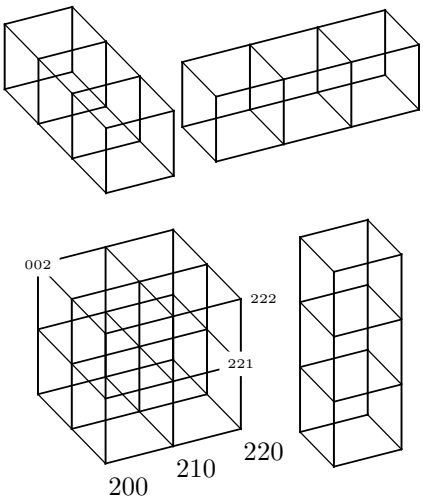


FIGURA 51: Figure 3D con `pst-gr3d`

Codice per la figura 51

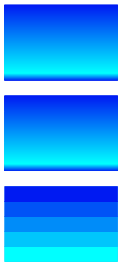
```
\PstGridThreeD(3,1,1)\quad
\PstGridThreeD(1,3,1)
\PstGridThreeD[GridThreeDNodes=true](2,2,2)
\SpecialCoor
\rput*(Gr3dNode002){\tiny 002}
\rput*(Gr3dNode221){\tiny 221}
\rput([Rx=0.3]Gr3dNode222){\tiny 222}
\multido{\i=0+1}{3}{%
\rput([Rx=0.3,Ry=-0.3]Gr3dNode2\i0){2\i0}}
\quad \PstGridThreeD(1,1,3)
```

pst-grad

Permette di generare sfumature di colore con più colori.

Codice per la figura 54

```
\newcommand{\Fig}[1][]{%
\begin{pspicture}(2,2)
\psframe[#1](2,2)
\end{pspicture}}
\newhsbcolor{ColorA}{0 0 0.7}
\newhsbcolor{ColorB}{0 1 0.7}
\newhsbcolor{ColorC}{.5 0.8 0}
\newhsbcolor{ColorD}{.5 0.8 1}
```



```
\begin{pspicture}(2,4)
\psset{fillstyle=gradient,
       linestyle=none}
\psframe[gradlines=5](1.5,1)
\psframe(0,1.2)(1.5,2.2)
\psframe[gradlines=1000]
        (0,2.4)(1.5,3.4)
\end{pspicture}
```

FIGURA 52: Si noti la qualità della sfumatura in relazione al parametro `gradlines`.

```
\begin{pspicture}(2,4)
\psset{fillstyle=gradient,
       linestyle=none,
       gradmidpoint=0.5}
\psframe[gradangle=0](1.5,1)
\psframe[gradangle=45](0,1.2)
                      (1.5,2.2)
\psframe[gradangle=90](0,2.4)
                      (1.5,3.4)
\end{pspicture}
```

FIGURA 53: Sfumature con diversa angolazione al variare del valore del parametro `gradangle`.

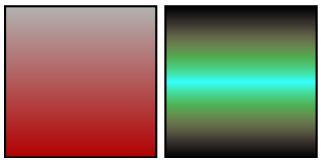


FIGURA 54: Sfumature con più colori.

```
\psset{fillstyle=gradient,
    gradientHSB=true}
\psframe[gradmidpoint=1,
gradbegin=ColorA,gradend=ColorB](2,2)
\psframe[gradmidpoint=0.5,
gradbegin=ColorC,gradend=ColorD](2,2)
```

pst-infixplot

Permette di disegnare grafici di funzioni a partire dalla notazione algebrica. Semplifica molto l'uso di `pst-plot` dal quale dipende e che carica automaticamente, insieme al pacchetto complementare `infix-RPN`.

Codice per la figura 55

```
\psaxes{->}(0,0)(0,-2)(5.5,3)
\infixtoRPN{sqrt(x)}
\psplot[linecolor=green]{0}{5}{\RPN}
\infixtoRPN{x^0.2}
\psplot[linecolor=red]{0}{5}{\RPN}
\infixtoRPN{sin(-x*180/3.1415)}
\psplot[linecolor=blue]{0}{5}{\RPN}
```

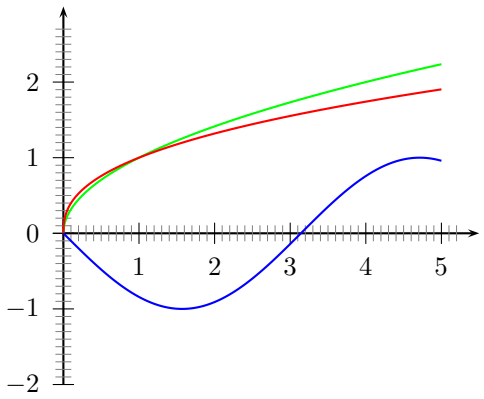


FIGURA 55: Grafici con `infix-RPN`.

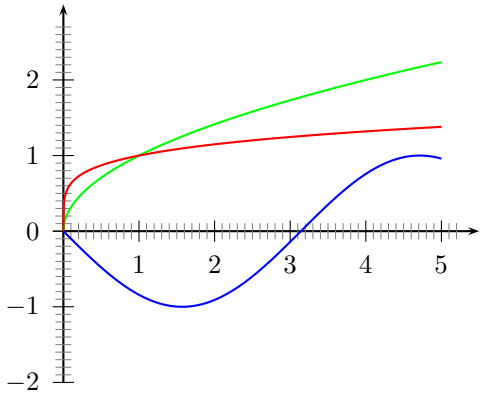


FIGURA 56: Grafici con `pst-infixplot`.

Codice per la figura 56

```
\psaxes{->}(0,0)(0,-2)(5.5,3)
\infixtoRPN{sqrt(x)}
\psPlot[linecolor=green]{0}{5}{sqrt(x)}
\psPlot[linecolor=red]{0}{5}{x^0.2}
\psPlot[linecolor=blue]{0}{5}
       {sin(-x*180/3.1415)}
```

pst-jtree

Consente di disegnare le strutture ad albero più usate nel campo della linguistica.

pst-labo

Consente di riprodurre le attrezzature di base di un laboratorio fisico-chimico. Di seguito sono proposte alcune immagini accompagnate dal codice che genera ciascun oggetto. Da notare l'uso combinato dei parametri.

Codice per la figura 57

```
\psset{bouchon=true}
\pstTubeEssais[glassType=tube]
\pstTubeEssais[glassType=ballon]
\pstTubeEssais[glassType=erlen]
\pstTubeEssais[glassType=flacon]
\pstTubeEssais[tubeDroit=true,
               tubePenche=10]
```

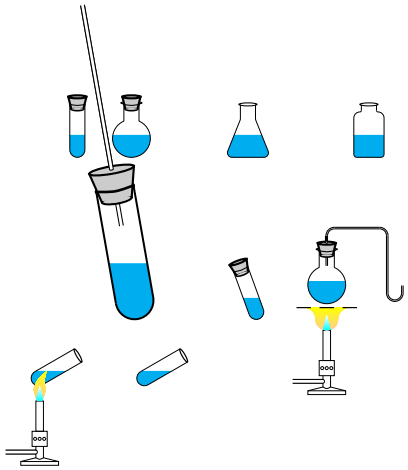


FIGURA 57: Teoria di strumenti da laboratorio.

```
\pstTubeEssais[tubePenche=20,bouchon]
\pstChauffageBallon[tubeRecourbeCourt]
\psset{tubeSeul=true}
\pstChauffageTube
\pstChauffageTube[becBunsen=false]
```

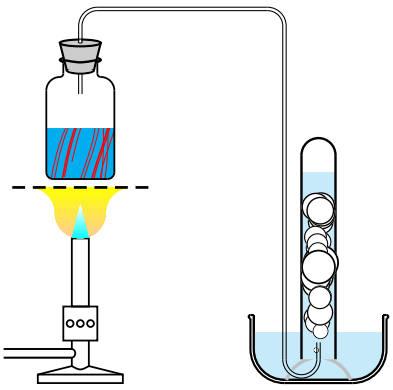


FIGURA 58: Ogni oggetto che compone la figura è *regolato* dall'uso dei parametri.

Codice per la figura 58

```
\psset{glassType=flacon,recuperationGaz,
       substance=\pstFilaments{red}}
\pstChauffageBallon[tubeRecourbe]
```

pst-lens

Implementa lenti per ingrandire o rimpicciolire figure, testo o altri oggetti di una pagina. Il codice seguente è quello della figura 17.

Codice per la figura 17

```
\def\Estremi{{ \rput[bl](0,0){%
\begin{pspicture}(0,0)(7,3)
\rput(3,1.7){\psline{o-oo}(-3,0)}
\rput[l](3.1,1.7){\texttt{o-o}}
```

```
\rput(3,1.5){\psline{*-**}(-3,0)}
\rput[l](3.1,1.5){\texttt{*-*}}
\rput(3,1.3){\psline{oo-oo}(-3,0)}
\rput[l](3.1,1.3){\texttt{oo-oo}}
\rput(3,1.1){\psline{**-**}(-3,0)}
\rput[l](3.1,1.1){\texttt{**-**}}
\rput(3,.9){\psline{c-c}(-3,0)}
\rput[l](3.1,.9){\texttt{c-c}}
\rput(3,.7){\psline{cc-cc}(-3,0)}
\rput[l](3.1,.7){\texttt{cc-cc}}
\rput(3,.5){\psline{C-C}(-3,0)}
\rput[l](3.1,.5){\texttt{C-C}}
\end{pspicture}
}}}
\begin{pspicture}(0,-1)(7,3.5)
\Estremi

\PstLens[LensHandle=false,
         LensSize=2.4,
         LensMagnification=3]
         (3.2,1.1){\Estremi}
\end{pspicture}
```

La lente ingrandisce quanto indicato nel secondo argomento del comando `\PstLens` con riferimento alle coordinate indicate nel primo argomento `(3.2,1.1)`. Risulta comodo definire una macro che definisce l'oggetto da sottoporre alla lente; in questo caso si è definita la macro `\Estremi`. I parametri usati indicano che non si vuole far disegnare il manico della lente: `LensHandle=false`, il raggio della lente è pari a 2.4 cm, l'ingrandimento è pari a tre volte l'originale. La lente può avere svariate forme personalizzabili.

pst-light3d

Implementa effetti di luce 3D

FIGURA 59: Nel codice riportato di seguito, si noti l'uso del parametro `LightThreeDAngle` che consente di variare l'angolo d'incidenza della luce.

Codice per la figura 59

```
\DeclareFixedFont{\Bf}{T1}{ptm}{b}
                          {n}{1.5cm}
\PstLightThreeDText[fillstyle=solid,
   fillcolor= white]{\Bf PSTricks}\\
\DeclareFixedFont{\Bf}{T1}{ptm}{b}
                          {n}{1.5cm}
\PstLightThreeDText[linestyle=none,
                    fillstyle=solid,
                  fillcolor=darkgray]
                          {\Bf Luce}\\
\psset{linestyle=none,fillstyle=solid,
                    fillcolor=green}
\PstLightThreeDText[LightThreeDAngle=0]
                              {\Bf Zero}\\
\PstLightThreeDText[LightThreeDAngle=90]
                             {\Bf Novanta}\\
```

pst-math

Estende, migliorandoli, gli operatori matematici standard. Il manuale del pacchetto è semplice e schematico, si possono ottenere grafici complessi con relativamente poco codice.

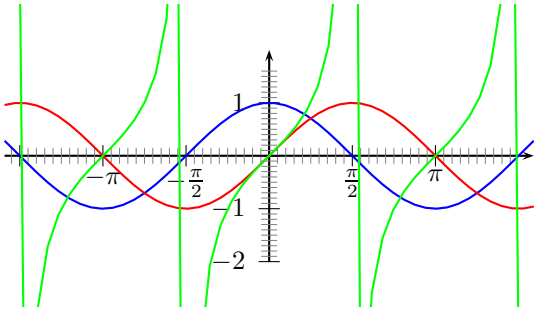


FIGURA 60: Esempio di grafico con il pacchetto `pst-math`.

Codice per la figura 60

```
\psset{unit=0.5cm,xunit=0.5cm,yunit=0.5cm}
\SpecialCoor
\psaxes[labels=y,Dx= \pstPI2]{->}%
(0,0)(-5,-2)(5,2)
\uput[-90](! PI 0){$\pi$}
\uput[-90](! PI neg 0){$-\pi$}
\uput[-90](! PI 2 div 0){$\frac{\pi}2$}
\uput[-90](! PI 2 div neg 0)%
{$-\frac{\pi}2$}
\psplot[linecolor=blue]{-5}{5}{x COS}
\psplot[linecolor=red]{-5}{5}{x SIN}
\psplot[linecolor=green]{-5}{5}{x TAN}
```

pst-node

Permette la costruzione di grafi aperti e chiusi con un completo controllo delle curve di collegamento con i nodi. L'idea di base consiste nell'assegnare un'etichetta ai nodi. Le etichette saranno gli estremi delle curve che collegano i nodi. Di seguito è riportato il codice della figura 2

Codice per la figura 2

```
\begin{psmatrix}[rowsep=.4cm,colsep=0.5cm]
& \rnode{Ed}{\psshadowbox[fillstyle=solid,
                         framesep=0.2]
             {Il vostro editor preferito}}\\
\rnode{ftex}{\psdblframebox{file.tex}}&
```

```
\rnode{ptex}{\psframebox{latex file.tex}}\\
\rnode{fdvi}{\psdblframebox{file.dvi}}&
\rnode{vdvi}{\rnode{vdvi}{%
\psframebox[fillcolor=blue,fillstyle=solid]
{\bfseries %
\color{white}Visualizzatore file DVI}}}\\
\rnode{pdvi}{\psframebox{dvips file.dvi}}&\\
\rnode{fps}{\psdblframebox{file.ps}}&
\rnode{vps}{\psframebox[fillcolor=red,
                        fillstyle=solid]
{\bfseries %
\color{white}Visualizzatore file PS}}\\
\rnode{ppspdf}{\psframebox{ps2pdf file.ps}}&\\
\rnode{fpdf}{\psdblframebox[doubleline=true]
                                    {file.pdf}}&
\rnode{vpdf}{\psframebox[framearc=.2,
                        fillstyle=gradient,%
                        gradangle=0,
                        gradbegin=blue,
                        gradend=green]%
{\bfseries
  \color{white}Visualizzatore file PDF}}
\nccurve[angleA=180,angleB=60,
           doubleline=true]{->}{Ed}{ftex}
  \nccurve[angleA=0,angleB=180]
  {->}{ftex}{ptex}
  \nccurve[angleA=245,angleB=60]
  {->}{ptex}{fdvi}
  \nccurve[angleA=0,angleB=180]
  {->}{fdvi}{vdvi}
  \nccurve[angleA=-90,angleB=90]
  {->}{fdvi}{pdvi}
  \nccurve[angleA=-90,angleB=90]
  {->}{pdvi}{fps}
  \nccurve[angleA=-90,angleB=90]
  {->}{pps}{fps}
  \nccurve[angleA=0,angleB=180]
  {->}{fps}{vps}
  \nccurve[angleA=-90,angleB=90]
  {->}{fps}{ppspdf}
  \nccurve[angleA=-90,angleB=90]
  {->}{ppspdf}{fpdf}
  \nccurve[angleA=0,angleB=180]
  {->}{fpdf}{vpdf}
  \end{psmatrix}
```

La sintassi è semplice, consideriamo la riga di codice:

`\rnode{ftex}{\psdblframebox{file.tex}}`

Il comando `\rnode` ha due argomenti, il primo è il nome del nodo `ftex`, il secondo è `\psdblframebox{file.tex}`; cioè un qualunque oggetto che vogliamo fare diventare il nodo del nostro grafo. A questo punto, individuati tutti i nodi di cui c'è bisogno, si creano le curve di collegamento:

```
\nccurve[angleA=0,angleB=180]
{->}{ftex}{ptex}
```

In sequenza: il nome del comando che genera la curva, i parametri che indicano l'angolo di partenza della curva e di arrivo sull'altro nodo. Il tipo di terminatore della curva, i nomi dei nodi di partenza e arrivo.

Il diagramma è stato costruito entro l'ambiente psmatrix. Si tratta di un ambiente semplice e flessibile, consente di creare semplici matrici di oggetti. Gli oggetti su ogni riga sono separati dal simbolo `&`. La fine riga è segnata dalle `\\`.

pst-ob3d

Implementa semplici oggetti grafici tridimensionali

pst-pdf

Utile strumento che consente di usare figure costruite con codice PSTricks in documenti PDF.

pst-optic

Per disegnare figure utili nell'illustrazione dell'ottica fisica e tecnica

pst-osci

Riproduce i grafici tipici generati dagli oscilloscopi.

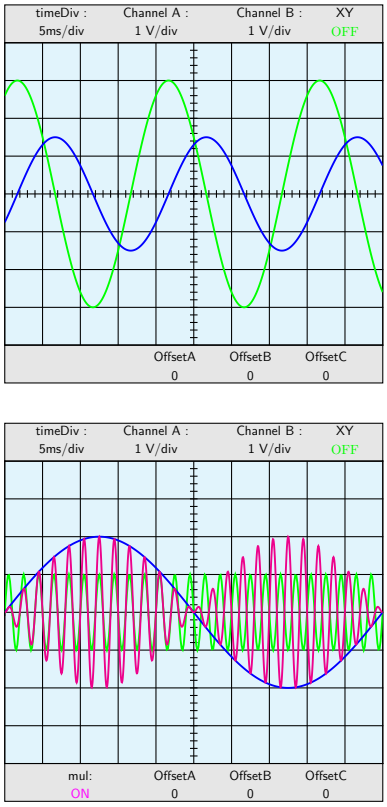


FIGURA 61: Oscilloscopio: `pst-osci`

Codice per la figura 61

```
\psscalebox{0.5}{\Oscillo[amplitude1=3,
              amplitude2=1.5, phase1=60,
              phase2=-30]}

\psscalebox{0.5}{\Oscillo[amplitude1=1,
                amplitude2=2,period2=50,
period1=2, combine= true, operation= mul]}
```

pst-pdgr

Consente di elaborare alberi genealogici utili in ambito medico.

pst-plot

Questo potente pacchetto definisce alcune macro che consentono di tracciare grafici a partire da un insieme di punti.

Le coordinate dei punti possono essere passate alla macro in due modi.

Primo metodo: consiste nell'inserire le coordinate dei punti in un file. A questo punto occorre indicare il file come argomento al comando che traccerà la curva.

```
\fileplot*[par]{filedati}
```

Il comando `\fileplot` consente di tracciare una curva, costruita per interpolazione, a partire dai punti elencati nel file `filedati` in termini di coordinate cartesiane. Il file dati dovrà essere composto con le seguenti regole:

- Tutti i dati devono essere contenuti fra parentesi quadre `[ ...]`
- I valori delle coordinate possono essere delimitati dalle parentesi graffe, dalle parentesi tonde, dalla virgola o da spazi.
- Non devono essere indicate unità di misura o altro.
- Sono ammessi i commenti identificati con il carattere %

```
[{{0, 0}, {1., 0.946083},
{2., 1.60541},{3., 1.84865},
{4., 1.7582},{5., 1.54993}}]
```

FIGURA 62: Esempio di file dati.

Secondo metodo: Con il `\dataplot` i dati possono essere inseriti come argomento del comando o passati dalla lettura di un file come nel caso precedente.

La sintassi è la seguente:

```
\dataplot*[par]{miocom}
```

I dati sono inseriti fra le parentesi quadre del comando:

```
\savedata*{miocom}[dati]
```

oppure letti da file con:

```
\readdata*{miocom}{nomefile}
```

Con `miocom` s'intende un nome generico scelto dall'utente che diviene a tutti gli effetti il comando `\miocom` col quale si richiamano i dati inseriti fra le parentesi quadre.

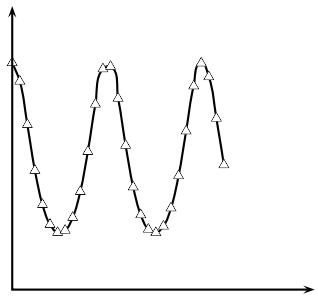


FIGURA 63: Esempio di grafico con `pst-plot`

Codice per la figura 63

```
\begin{pspicture}(-1.5,0)(6,5)
\psset{xunit=.2cm,yunit=1.5cm}
\savedata{\ImieiDati}[{%
{0,2},{0.5,1.837294076},
{1,1.454277218},{1.5,1.050253215},
{2,0.749423521},{2.5,0.573894075},
{3,0.503480394},{3.5,0.522514621},
{4,0.635672851},{4.5,0.864060479},
{5,1.217280953},{5.5,1.634296532},
{6,1.945539521},{6.5,1.967805494},
{7,1.686347963},{7.5,1.271591535},
{8,0.904065977},{8.5,0.65883454},
{9,0.531768321},{9.5,0.500981017},
{10,0.559003209},{10.5,0.719199076},
{11,1.00307237},{11.5,1.397942246},
{12,1.794838404},{12.5,1.996950095},
{13,1.875722987},{13.5,1.510389524},
{14,1.099415876}}]
\dataplot[plotstyle=curve,showpoints=true,
   dotstyle=triangle]{\ImieiDati}
\psline{<->}(0,2.5)(0,0)(20,0)
\end{pspicture}
```

Esiste un altro comando per disegnare un grafico a partire dalle coordinate:

```
\listplot*[par]{lista}
```

la sostanziale differenza con gli altri, è che in questo caso, la lista di coordinate deve essere separata solo degli spazi.

Per tracciare il grafico di una generica funzione $y = f(x)$ esiste il comando:

$$\texttt{\textbackslash psplot*[par]\{}x_{min}\texttt{\}\{}x_{max}\texttt{\}\{}f(x)\texttt{\}}$$

La funzione $f(x)$ deve essere espressa in codice PostScript.

Codice per la figura 64

```
\begin{pspicture}(0,-2)(5,2)
\psset{xunit=0.2pt}
\psplot[linecolor=red,plotstyle=curve]%
{0}{720}{x cos }
\psplot[plotpoints=100]{0}{720}{x sin }
\psline{<->}(0,-1.5)(0,1.5)
```

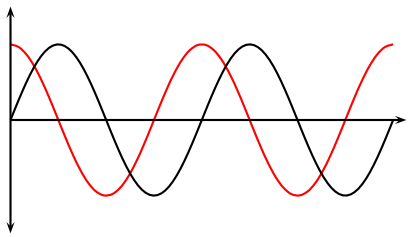


FIGURA 64: Esempio di grafico con `psplot`.

```
\psline{->}(745,0)
\end{pspicture}
```

È inoltre possibile disegnare i grafici di funzioni parametriche con il comando:

`\parametricplot*[par]{`x_{min}`}{`x_{max}`}{`$x(t)\ y(t)$`}`

Anche in questo caso la funzione parametrica deve essere definita tramite codice PostScript.

pst-poly

Consente di disegnare poligoni regolari e non.

pst-slpe

Questo pacchetto permette di generare gradienti di colore in modo efficiente superando le limitazioni del pacchetto pst-grad.

pst-stru

Consente di disegnare gli schemi tipici delle strutture semplificate usate nell'ambito della scienza delle costruzioni.

pst-text

Consente di manipolare le parole di un testo per ottenere svariati effetti grafici. Le parole seguono la curva che è costruita per interpolazione attraverso i punti indicati in coordinate cartesiane oppure tracciata con un comando standard. Nel primo esempio di figura 65 la curva è un cerchio creato con il comando standard `\pscircle`. Il parametro `linecolor=white` assegna il colore alla curva *guida*. Nel secondo esempio `linecolor=gray`.

pst-tree

Consente di generare strutture ad albero con diversi stili.

pst-uml

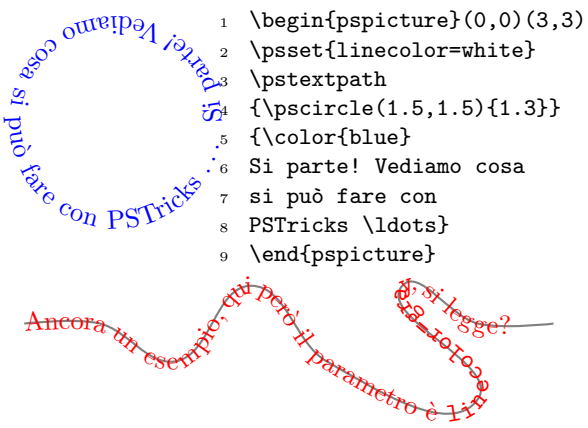
Consente di disegnare diagrammi in notazione UML.

pst-vue3d

Consente di rappresentare e visualizzare oggetti in 3D.

bardiag

Consente di disegnare istogrammi.




```
\begin{pspicture}(0,0)(3,3)
\psset{linecolor=white}
\pstextpath
{\pscircle(1.5,1.5){1.3}}
{\color{blue}
Si parte! Vediamo cosa
si può fare con
PSTricks \ldots}
\end{pspicture}
```

```
\begin{pspicture}(0,-1)(7,.7)
\psset{linecolor=gray}
\pstextpath
{\pscurve(0,0)(1,0)(2,-.5)(3,.5)(4,-.5)(6,-1)
(5,.5)(6,0)(7,0)}
{\color{red}
Ancora un esempio, qui però il parametro è
\texttt{linecolor=gray}, si legge?}
\end{pspicture}
```

FIGURA 65: Esempio per `ps-text`.

dbicons

Permette di generare la struttura di data base (ER-diagrams).

euklides

Consente di generare figure di geometria piana.

gastex

Collezione di macro per generare disegni, diagrammi e figure di diverso tipo.

JasTeX

Interfaccia sviluppata in Java per GasTeX.

makeplot

Consente di disegnare grafici a partire da dati generati con Mathlab.

multido

Macro utile per generare operazioni ripetitive.

psgo

Consente di disegnare diagrammi inerenti il gioco GO.

RRGtree

Alberi sinttatici basati su RRG (Role and Reference Grammar).

spectrum

Consente di generare e gestire colori per produrre figure e testo con sfumature complesse.

uml

Un altro pacchetto per generare diagrammi UML.

vaucanson

Pacchetto per la generazione di grafi.

Sketch

Sistema per generare figure in 3D.

psMath

Sistema per generare figure bi e tridimensionali.

Riferimenti bibliografici

Internet è senz'altro la fonte principale dove reperire le informazioni su PSTricks. Nessun sito contiene tutto. Non tutti i pacchetti sono inseriti nelle distribuzioni ufficiali di TEX e LATEX.

URL `http://tug.org/PSTricks/main.cgi/`. Questo sito contiene tutta la documentazione e gli aggiornamenti del pacchetto base e della maggior parte dei pacchetti correlati. Vi si trovano molti altri collegamenti ad altre risorse. Di seguito altri siti utili. `ftp://ftp.ctan.org/tex-archive/graphics/pstricks/` `http://melusine.eu.org/syracuse/` `http://members.aol.com/mluque5130/` `http://latexdraw.sourceforge.net/index.html`.

Caschili, M. (2006). «Semplici figure con l'ambiente picture». *ArsTEXnica*, (1), pp. 20–28. URL `http://www.guit.sssup.it/arstexnica.php`.

Goossens, M., Mittelbach, F., Rahtz, S., Roegel, D. B. e Voss, H. (2007). *The LATEX Graphics Companion*. Tools and Techniques for Computer Typesetting. Addison-Wesley Professional, Reading, Massachusetts, 2ª edizione.

Voss, H. (2007). *PSTricks: Grafik für TEX und LATEX*. Pubblicato nel febbraio del 2007. A breve è prevista l'uscita della traduzione in lingua inglese.

▷ Massimo Caschili
`massimo.caschili@tiscali.it`

LaTeX nella Scuola Media Superiore: applicazioni didattiche con PSTricks

Luciano Battaia

Sommario

Scopo del presente articolo è discutere la possibilità di introdurre, con positive e importanti ricadute didattiche, l'uso di LaTeX, e in particolare del pacchetto PSTricks con le sue estensioni, nella scuola media superiore.

L'articolo, che trae origine da un'esperienza fatta negli ultimi anni presso il Liceo Scientifico Grigoletti di Pordenone e principalmente in una classe terza dell'anno scolastico 2006/2007, non riporta i dettagli d'uso dei pacchetti LaTeX citati, quanto piuttosto una serie di idee su come riuscire a conciliare LaTeX (e in particolare PSTricks) con i normali (e straobsoleti) dettami dei programmi scolastici, in modo da proporre qualcosa che sia interessante per gli studenti e utile per il loro futuro di "utenti LaTeX".

Abstract

The aim of this paper is to promote the use of LaTeX, and in particular PSTricks with its extensions, in secondary high school and to debate the positive influences of this fact on math teaching.

The paper is not a technical introduction to PSTricks: rather it reflects the author's ideas concerning the real possibility of reconciling both "old fashioned" school programs and LaTeX strategies. The techniques discussed here have been used in a course at Liceo Scientifico Grigoletti, Pordenone.

1 Introduzione

La diffusione dei software di tipo WYSIWYG di videocomposizione di testi, favorita anche dalla facilità con cui erano, e in parte sono tuttora, reperibili copie contraffatte degli applicativi più diffusi, ha fatto proliferare a dismisura, anche nella scuola media superiore, gli "appunti" autoprodotti distribuiti dai docenti, spesso anche in sostituzione dei libri di testo. Con questo è venuta a mancare l'operazione di "filtro" che eseguivano i tipografi professionisti prima di stampare e diffondere un testo. Ciò ha provocato risultati, a mio parere, disastrosi, e non parlo qui dei contenuti, per i quali forse varrebbe comunque la pena di spendere alcune parole, ma della forma.

Riporto, a titolo d'esempio, il testo di un esercizio in preparazione ai test per l'ammissione a una scuola di eccellenza, pubblicato sul sito ufficiale di una università italiana:

> Col simbolo [x] si indica la parte intera del numero reale x, ossia il più grande intero che non supera x. Trovare tutte le soluzioni dell'equazione $x[x[x]]=82$.

Il grosso problema è che tutto questo ha fatto perdere il "gusto" per i documenti ben formati, piacevoli da consultare, e quindi da studiare, ponendo principalmente l'attenzione sulle enormi possibilità di adattamenti puramente estetici, spesso privi di ogni senso. In una tesina dei recenti esami di stato (valutata con il massimo dei voti da parte dei commissari), ho potuto contare ben *dodici* font diversi utilizzati nelle varie parti (compreso naturalmente l'onnipresente Comic Sans).

In questa situazione tentare di diffondere la passione per un software come LaTeX è un'impresa abbastanza ardua: l'impegno richiesto per produrre risultati, anche se di indubbia qualità, è troppo elevato e l'opinione diffusa è che il gioco non valga la candela. Per esempio uno dei commenti che ho spesso sentito è che, in fondo, non è molto grave se in uno stesso documento trovo scritto a volte $\mathrm{arc}tgx$, a volte $\mathrm{arctg}\,x$, a volte $arctgx$.

Naturalmente questo non significa che si debba rinunciare a "fare proseliti", ma solo che bisogna affrontare il problema da più angolature. Introdotti i rudimenti della costruzione di un documento LaTeX, e mostrato che in sostanza non c'è nulla di diverso rispetto ad un normale word processor, è opportuno proporre subito lavori che possono essere realizzati *solo* in LaTeX. Se poi si riesce a inserire l'apprendimento delle tecniche avanzate di videoscrittura all'interno delle normali attività didattiche, come un normale esercizio sugli argomenti che si stanno trattando, allora, forse, il gioco è fatto.

I docenti di materie scientifiche sono chiaramente facilitati in questo: in fondo LaTeX è nato proprio per loro, ma, secondo me, altre interazioni sono possibili.

Propongo qui la mia esperienza, basata appunto sull'idea di inserire l'apprendimento di LaTeX e dei pacchetti connessi all'interno delle normali attività didattiche, esattamente come un esercizio applicativo nei corsi di matematica.

In questo articolo parlo solo di applicazioni alla geometria euclidea, ma PSTricks può fare molto

di più: chi è interessato ad approfondirne la conoscenza può visitare il sito web ufficiale, contenente tra l'altro una estesa raccolta di esempi completi di codice sorgente, all'indirizzo `http://tug.org/PSTricks/main.cgi/`. Segnalo inoltre che la seconda edizione del *Latex Graphics Companion*, uscita da pochissimo, contiene ben 250 pagine dedicate proprio a PSTricks.

2 Costruzioni geometriche con PSTricks

Una delle necessità più comuni per chi deve produrre appunti di matematica è quella dell'inserimento di grafici e di costruzioni geometriche. Esistono numerosi software, anche free, in grado di soddisfare queste esigenze, ma a quanto mi risulta moltissimi colleghi usano software di geometria dinamica, tipo CABRI[1], per ottenere le immagini richieste, splendide per la visualizzazione su schermo, ma alquanto povere per quanto riguarda la resa tipografica.

Il pacchetto LaTeX che, secondo me, consente i risultati migliori in questo campo è PSTricks con le sue varie e numerose (forse troppo) estensioni. Esistono altre soluzioni, tra cui è doveroso citare pgf, per certi versi, in particolare la sua possibilità di scrivere direttamente il codice pdf nel file di uscita, decisamente interessante. Si potrebbe anche tentare di usare direttamente Metapost, ma la mia idea è quella di rimanere, per quanto possibile, nell'ambito delle conoscenze matematiche di uno studente di terza liceo, tentando anzi di usare la scrittura di codici come esercizio di matematica.

Per chiarire il senso di quello che voglio dire, riporto uno dei primi esercizi che propongo in classe: costruire, e disegnare, un triangolo $\triangle ABC$ di cui siano dati i lati $AB = 6\,\text{cm}$, $BC = 4\,\text{cm}$, $AC = 5\,\text{cm}$. Poiché PSTricks usa le coordinate cartesiane per posizionare gli oggetti nella pagina, disponiamo opportunamente il triangolo in modo da semplificare la costruzione della figura. Scegliamo di mettere il punto A nell'origine degli assi cioè in $(0,0)$, il punto B in $(6,0)$ e il punto C nel primo quadrante, in una posizione da determinare, vedi la figura 1.

Il modo più semplice per determinare la posizione di C è quello di usare le coordinate polari: $C = (\rho;\theta)$ (notazione propria di PSTricks). Si ha naturalmente $\rho = \overline{AC} = 5$. Per trovare θ, ovvero l'angolo $C\widehat{A}B$, usiamo la formula trigonometrica che coinvolge l'arcotangente. Non si tratta dell'unica formula possibile, ma è da preferire in questo caso in quanto il linguaggio PostScript ha l'arcotangente tra le funzioni predefinite.

1. Citerò spesso in questo articolo i software di geometria dinamica. Farò normalmente riferimento a CABRI solo perché è quello in uso nella mia scuola e tra quelli più diffusi, ma le osservazioni fatte si applicano a uno qualunque dei software in commercio, compresi alcuni esempi free estremamente interessanti, tra cui è d'obbligo di citare KIG, reperibile all'indirizzo `http://edu.kde.org/kig/`.

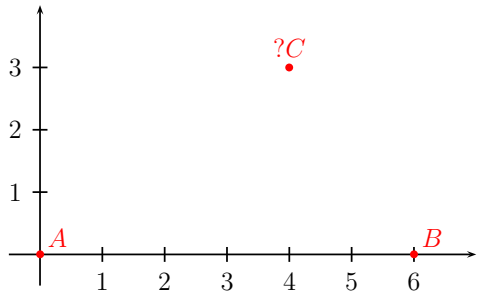


FIGURA 1: Costruzione di un triangolo dati i lati

La formula richiesta è:

$$C\widehat{A}B = 2\arctan\sqrt{\frac{(s-b)(s-c)}{s(s-a)}},$$

ove s è il semiperimetro e a, b, c sono i lati opposti agli angoli $\widehat{A}, \widehat{B}, \widehat{C}$, come d'uso. I valori adatti al nostro problema sono: $s = 7.5$, $s-a = 3.5$, $s-b = 2.5$, $s-c = 1.5$. Dovremo allora calcolare

$$2\arctan\sqrt{\frac{2.5\times 1.5}{7.5\times 3.5}}.$$

Si potrebbe usare una calcolatrice e inserire direttamente il suo risultato nel codice LaTeX, ma considerato lo scopo di questo esercizio è preferibile inserire direttamente il codice PostScript per eseguire il calcolo.

A proposito di PostScript mi pare interessante, sempre dal punto di vista didattico, l'uso della notazione polacca inversa che questo linguaggio richiede, in particolare la necessità di scrivere le formule senza parentesi: si tratta di un ottimo esercizio per costringere lo studente a ragionare sull'ordine delle operazioni, sul significato delle proprietà delle operazioni stesse, sul modo con cui, a livello elementare, vengono eseguite le varie istruzioni in un calcolatore automatico. Per fare un esempio elementare, gli studenti hanno trovato molto significativo il fatto che le due notazioni possibili, in RPN, per $a+b+c$, *a b add c add* oppure *a b c add add*, corrispondano effettivamente a due diverse successioni di operazioni del processore, che portano allo stesso risultato solo per la (troppo spesso sottovalutata) proprietà associativa.

Riporto, per completezza, il codice definitivo della costruzione:

```
\documentclass{minimal}
\usepackage{pstricks}
\begin{document}
\thispagestyle{empty}
\begin{pspicture}(0,-0.5)(6,4)
\SpecialCoor
\pspolygon[linecolor=blue](0,0)(6,0)%
```

```
  (!2.5 1.5 mul sqrt 7.5 3.5 mul sqrt%
  atan 2 mul cos 5 mul 2.5 1.5 mul sqrt%
  7.5 3.5 mul sqrt atan 2 mul sin 5 mul)
\end{pspicture}
\end{document}
```

L'esercizio può essere reso ancora più interessante se si richiede, utilizzando il minimo insieme di macro PSTricks, di piazzare i nomi sui vertici, le lunghezze sui lati, di marcare gli angoli, ecc., risultato che si può vedere nella figura 2.

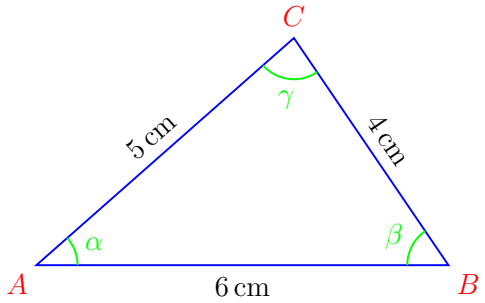


FIGURA 2: Costruzione di un triangolo dati i lati: la figura finale

Per chi fosse interessato a conoscere tutti i dettagli, un articolo esaustivo si può reperire in BATTAIA (2005).

La ricaduta didattica di un lavoro di questo tipo è notevole: anche gli studenti più distratti sono stimolati a apprendere i numerosi concetti di geometria analitica e trigonometria necessari al completamento della figura. Ritengo che pratiche di questo tipo siano ben più produttive di decine di esercizi "tradizionali", con il risultato secondario (ma ovviamente molto importante) di avere preso dimestichezza con il sistema LaTeX.

3 Il pacchetto pst-eucl

Un approccio completamente diverso, e ugualmente interessante per le sue ricadute didattiche, si ottiene con l'uso del pacchetto pst-eucl, una delle decine di estensioni di PSTricks. Esso mette a disposizione dell'utente tutte le ordinarie tecniche di costruzione "con riga e compasso" della geometria euclidea, oltre a consentire una facile manipolazione degli strumenti tipici di alcune trasformazioni affini del piano in sé. Si hanno dunque a disposizione strumenti molto più sofisticati, che nascondono in parte gli algoritmi di base utilizzati, ma che consentono importanti e promettenti sviluppi.

Per quanto riguarda il problema in discussione, si può utilmente far vedere agli studenti come la costruzione richiesta si semplifichi se si hanno a disposizione gli strumenti ordinari del geometra, senza perdere di vista la bellezza di un risultato ottenuto con tecniche elementari. Nel dettaglio si può ora determinare la posizione del punto C esattamente come si fa su un foglio di carta o con un software di geometria dinamica come CABRI: dati i punti A e B come nodi, si tratta di trovare una delle due intersezioni tra le circonferenze di centro A e raggio 5 e di centro B e raggio 4. Il codice è elementare e, soprattutto, estremamente intuitivo. Richiamato il pacchetto pst-eucl nel preambolo, il codice precedente si modifica in:

```
\begin{pspicture}(0,-0.5)(6,4)
\pstGeonode(0,0){A}(6,0){B}
\pstInterCC[RadiusA=\pstDistVal{5},%
  RadiusB=\pstDistVal{4}]%
  {A}{}{B}{}{C'}{C''}
\pspolygon[linecolor=blue](A)(B)(C')
\end{pspicture}
```

La cosa estremamente interessante di questo pacchetto, per gli scopi che stiamo perseguendo, è che opera esattamente come CABRI: nelle varie costruzioni si devono fissare alcuni punti base (i "nodi"), e tutto il resto è realizzato direttamente dal codice con gli strumenti classici della geometria. Questo consente di costruire con grande facilità diverse versioni della stessa figura. Considerate poi le affinità con il metodo di lavoro di CABRI, è possibile sfruttare le capacità dinamiche di questo software per costruire bozze di figure che poi potranno essere realizzate con un codice LaTeX da inserire direttamente in un documento che le richieda.[2]

4 Inversioni circolari

Una volta che gli studenti si sono impratichiti con i primi semplici esercizi di costruzioni con PSTricks, è venuto il momento di passare a costruzioni più complesse. La scelta che ho fatto nell'anno scolastico 2006/2007 è stata quella di trattare la trasformazione per raggi vettori reciproci (inversioni circolari) e in particolare alcuni classici problemi connessi, principalmente il *problema di Apollonio*, ma anche il *porisma di Steiner* o costruzioni relative all'*arbelo*.

Le inversioni circolari costituiscono un argomento raramente trattato a livello di scuola media superiore e, a quanto mi risulta, anche a livello di primi anni di Università. Eppure lo strumento dell'inversione è decisivo nella risoluzione di numerosi problemi e, in altri casi, rende la trattazione semplice ed elegante. Si può aggiungere che, se si rinuncia a entrare troppo nei dettagli tecnici, non si tratta di un problema teoricamente complesso e sicuramente alla portata di uno studente medio.

2. PSTricks e i pacchetti connessi producono codice PostScript che non può essere inserito direttamente in un file da compilare con PDFLaTeX. Fino a qualche tempo fa era possibile solo il solito passaggio *dvi→ps→pdf*, o l'inserimento di immagini esterne. Ora esistono anche dei pacchetti (come pst-pdf) che rendono praticamente automatica la compilazione anche con PDFLaTeX di file contenente codice PSTricks.

L'uso poi di CABRI e, nel nostro caso, di PSTricks rende ancora più affascinante la trattazione.

Anche in questo caso le interazioni con CABRI sono state molto importanti. Per esempio la costruzione delle circonferenze che risolvono il *problema di Apollonio* è, quasi sempre, tecnicamente complessa e richiede una attenta analisi delle situazioni di partenza per ottenere risultati significativi ed esteticamente accettabili: molto spesso le costruzioni debordano facilmente dal foglio di disegno e si rischia di produrre un grafico bellissimo, ma magari visualizzabile solo in un foglio di formato A0! L'uso di un software di geometria dinamica, come CABRI, può grandemente semplificare il problema, in quanto permette di modificare velocemente i dati iniziali fino a scegliere una situazione adatta ad essere trasferita efficacemente su carta.

Il primo problema che si è presentato è stato quello di costruire delle macro per velocizzare alcuni passaggi che si ripetono molto spesso nelle costruzioni richieste. La scelta che ho fatto è stata quella di ricercare codici semplici e facilmente costruibili anche da parte di chi non ha particolare dimestichezza con LaTeX: gli studenti destinatari del progetto avevano fatto non più di $6-7$ lezioni sull'argomento.

Ovviamente la macro più importante è proprio quella che costruisce l'inverso di un punto, dato il centro e il raggio del circolo di inversione. Riporto qui di seguito quella che abbiamo utilizzato (e che sicuramente potrebbe essere perfezionata da parte di qualche esperto!).

```
\makeatletter
\newcommand{\pstInv}[5][]{%
\pstInterLC[Radius=\pstDistVal{#3},%
  PointNameA=none,PointSymbolA=none,%
  PointNameB=none,PointSymbolB=none]%
  {#2}{#4}{#2}{}{@X}{@Q}%
\pstRotation[RotAngle=90,PointName=none,%
  PointSymbol=none]{#2}{@Q}[@T]%
\pstTranslation[PointName=none,%
  PointSymbol=none]{#4}{@Q}{@T}[@S]%
\pstInterLL[PointName=none,%
  PointSymbol=none]{@Q}{@S}{#2}{@T}{@N}%
\pstCircleOA[linestyle=none]{#2}{@N}%
{\psset{PointNameA=none,%
  PointSymbolA=none,PointNameB=none,%
  PointSymbolB=none}%
\pstInterLC[#1]{#2}{#4}{#2}{@N}{@R}{#5}}%
}
\makeatother
```

Il comando `\pstInv`, così definito, richiede 5 parametri, di cui uno opzionale.

- La macro non disegna di default il punto inverso, né il suo nome. Nel parametro opzionale si possono inserire le opzioni per modificare questo comportamento, e precisamente:
  - `PointSymbolB=default` (o qualcos'altro) per disegnare il punto inverso;
  - `PointNameB=default` (o qualcos'altro) per scrivere il nome del punto inverso;
  - altre opzioni tipiche di pst-eucl (per esempio `PosAngle`).
- Il primo parametro obbligatorio è il centro di inversione (un nodo già definito).
- Il secondo parametro obbligatorio è il raggio del cerchio di inversione.
- Il terzo parametro obbligatorio è il punto di cui calcolare l'inverso (un nodo già definito).
- Il quarto parametro obbligatorio è il nome di nodo del punto inverso.

A questo punto naturalmente si è dovuto procedere allo studio analitico dei problemi proposti. In tutti i casi abbiamo prima sperimentato le costruzioni con CABRI e successivamente siamo passati alla traduzione dei vari passaggi nel codice PSTricks.

Il lavoro è stato abbastanza complesso e i codici delle varie figure possono essere anche lunghi un centinaio di righe, ma devo concludere che i risultati sono stati ritenuti molto soddisfacenti da parte degli studenti e che la maggior parte di loro è riuscita a svolgere agevolmente i compiti assegnati. Tutto il materiale, comprensivo dei codici completi, è consultabile in BATTAIA (2007a) e BATTAIA (2007b).

A mio avviso è abbastanza interessante il fatto che siamo riusciti a produrre codici facilmente configurabili per trattare diversi casi. Per esempio nel caso del *porisma di Steiner*, le varie situazioni possono essere trattate semplicemente modificando alcuni parametri posti nella parte iniziale del codice:

`\newcommand{\rgrande}{`⟨raggio del cerchio esterno⟩`}`
`\newcommand{\distcentri}{`⟨distanza tra i due centri dei cerchi base⟩`}`
`\newcommand{\numcirc}{`⟨numero di cerchi della catena⟩`}`
`\newcommand{\anginiz}{`⟨angolo (espresso in gradi) del primo centro⟩`}`

La cosa permette di costruire facilmente delle animazioni visualizzabili in Acrobat se si ha la versione almeno 7. Le animazioni possono essere realizzate con estrema semplicità utilizzando il nuovo pacchetto animate di Alexander Grahn.

Riporto per completezza nelle figure 3 e 4 l'immagine finale della costruzione dei cerchi di Apollonio, nel caso più complesso delle circonferenze tangenti a tre circonferenze date, e una delle immagini relative al *porisma di Steiner*.

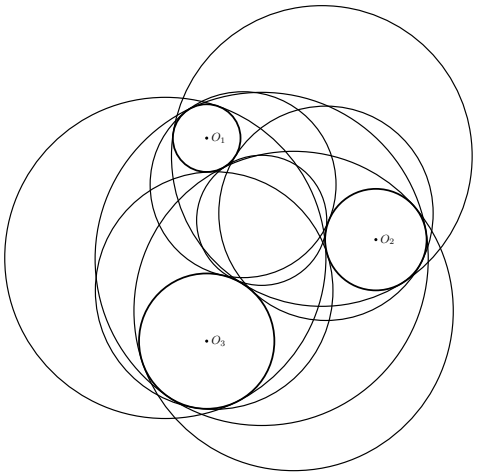


FIGURA 3: Il problema di Apollonio: circonferenze tangenti a tre circonferenze date

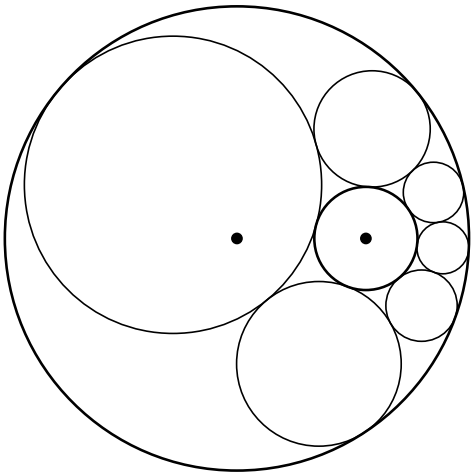


FIGURA 4: Il porisma di Steiner: una catena con 6 circonferenze

5 Un compito per casa

Oltre ai lavori principali realizzati in classe, nei quali l'aspetto "matematico" era preminente, sono stati assegnati numerosi lavori domestici agli studenti. Tra questi uno è preso da un'idea pubblicata da Martin Gardner in *The Sixth Book of Mathematical Games from Scientific American*, Chicago, IL, University of Chicago Press, 1984, e si riferisce alla costruzione dell'inversa di una normale scacchiera, rispetto ad un circolo di inversione con centro nel centro della scacchiera base.

Si tratta di una costruzione semplice nella sua impostazione generale, ma assai complessa nella sua realizzazione tecnica, a causa dei numerosi passaggi richiesti. La soluzione che propongo qui, vedi la figura 5, è stata realizzata interamente da uno studente di 3ª Liceo Scientifico[3], con solo limitatissimi aiuti da parte del docente. Poiché il codice completo contiene alcune centinaia di righe, mi pare un risultato di indubbio valore. Tra l'altro questa volta PSTricks si è rivelato superiore a CABRI, con il quale la colorazione delle varie zone non si rivela affatto banale.

3. Lorenzo de Francesco, 3ªB, Liceo Scientifico "Grigoletti" di Pordenone.

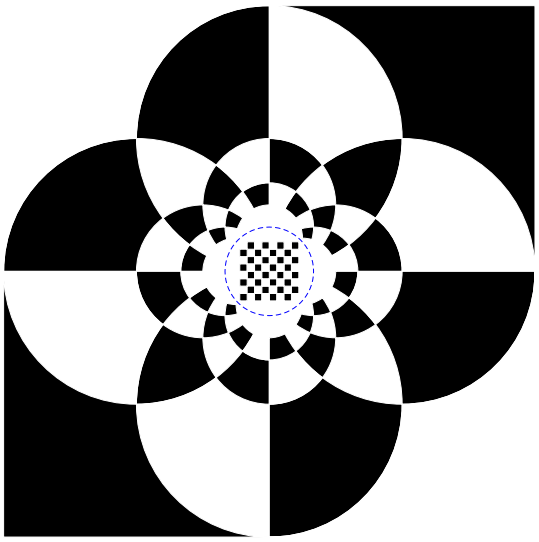


FIGURA 5: Una normale scacchiera assieme alla sua inversa

6 Idee per il futuro

Visto il successo della modalità di approccio a LaTeX che ho proposto in questo articolo, intendo continuare anche per il futuro, cercando magari ulteriori strategie per interessare studenti, e docenti, anche non dell'ambito strettamente scientifico.

Uno dei progetti che conto di realizzare è quello di coinvolgere i docenti di lettere e lingua straniera, per esempio usando i pacchetti gb4e e cgloss4e di Hans-Peter Kolb e Craig Thiersch per traduzioni interlineari o altre opportunità offerte dai pacchetti stessi.

Il tentativo di coinvolgere nell'uso di LaTeX docenti che "non hanno bisogno di formule o grafici" non è semplice: un uso accurato di un programma di videoscrittura WYSIWYG[4] consente di ottenere in genere risultati abbastanza soddisfacenti. I colleghi che ho contattato sono però rimasti molto impressionati dalla facilità con cui si può produrre in LaTeX un pezzo di documento come quello della figura 6.

Se son rose, fioriranno,...

7 Conclusioni

I risultati del progetto di introduzione a LaTeX nella scuola media superiore, che ho qui descrit-

4. In realtà la quasi totalità degli utenti che ho avuto modo di incontrare usa questo tipo di programmi solo come una macchina da scrivere un po' sofisticata, e non conosce nemmeno tutte le varie opzioni di "progettazione" dei documenti che vengono fornite.

Ne quidem nostris temporibus
Nemmeno nei nostri tempi
aetas quamquam incuriosa
la generazione sebbene incurante
suorum omisit tradere, antiquitus
dei suoi tralasciò di affidare, anticamente
usitatum, posteris facta et mores
abituale, ai posteri le opere e i costumi
virorum clarorum, quotiens
degli uomini illustri, ogni volta che
aliqua magna ac nobilis virtus vicit ac
una grande e nobile virtù vinse e
supergressa est vitium commune
superò il vizio comune
parvis et magnis civitatibus,
alle piccole e alle grandi città,
ignorantiam recti et invidiam.
l'ignoranza del bene e l'invidia.

FIGURA 6: Tacito: *Vita di Agricola*, con traduzione interlineare

to, sono molto incoraggianti e spero ci potranno essere ulteriori sviluppi nel futuro.

Purtroppo, come osserva Peter Flynn in *Formatting Information*, Silmaril Consultants, «Molti sono diventati "Drogati del Word Processing" e non *scrivono* documenti, li *disegnano*, quasi nello stesso modo in cui un bambino di tre anni, in età prescolare, potrebbe fingere di "scrivere" una storia, mentre sta solo creando una sequenza di simboli con un blocco di carta e una scatola di pennarelli». Tentare di passare l'idea che bisognerebbe innanzitutto preoccuparsi di *scrivere* un documento, piuttosto che perdere una gran parte del tempo a fare un editing casuale e a "giocare con il mouse", è un compito arduo e impegnativo e spesso i vincoli imposti dal "sistema LaTeX" sono mal sopportati se non detestati. Recentemente ho avuto uno scambio epistolare con uno studente impegnato a scrivere la propria tesi: aveva utilizzato la classe toptesi e non riusciva a mandare giù il fatto che, dopo tutta la fatica fatta, non riusciva a ingrandire i caratteri come voleva, in modo da guadagnare qualche pagina!

In ogni caso mi auguro di avere contribuito a diffondere la convinzione che fare qualcosa per estendere l'interesse per i documenti ben fatti è ancora possibile, anche nella scuola media superiore.

Riferimenti bibliografici

BATTAIA, L. (2005). «Triangoli, PSTricks e Cabri». Disponibile su `www.batmath.it/latex/pdfs/costr_tri.pdf`.

— (2007a). «Apollonio, inversioni e PSTricks». Disponibile su `www.batmath.it/latex/pdfs/apollonio_pst.pdf`.

— (2007b). «Il porisma di Steiner». Disponibile su `www.batmath.it/latex/pdfs/steiner.pdf`.

BECCARI, C. (2005). «La classe toptesi». Disponibile su CTAN `macros/latex/contrib/toptesi`.

GRAHN, A. (2007). «The animate package». Disponibile su CTAN `macros/latex/contrib/animate`.

KOLB, H.-P. e THIERSCH, C. (2006(?)). «The gb4e and cgloss4e packages». Disponibile su CTAN `macros/latex/contrib/gb4e`.

NIESPRACHK, R. e GÄSSLEIN, H. (2006). «The pst-pdf package». Disponibile su CTAN `graphics/pstricks/contrib/pst-pdf`.

RODRIGUEZ, D. (2005). «The pst-eucl package». Disponibile su CTAN `graphics/pstricks/contrib/pst-eucl`.

TANTAU, T. (2006). «The TikZ and pgf packages». Disponibile su CTAN `graphics/pgf`.

VAN ZANDT, T. (2003). «PSTricks: Postscript macros for generic TeX». Disponibile su CTAN `graphics/pstricks/base/generic`.

▷ Luciano Battaia
Liceo S. "Grigoletti" - Pordenone
batmath@gmail.com

Illustrazioni tridimensionali con Sketch/LaTeX/PSTricks/Ti*k*z nella didattica della Dinamica del Volo

A. De Marco

Sommario

In questo articolo viene mostrato come sia possibile utilizzare le potenzialità di LaTeX e dei pacchetti PSTricks e Ti*k*z nella produzione di illustrazioni avanzate. La creazione di disegni che rappresentino delle scene tridimensionali con indicazioni di contenuto scientifico è di fatto possibile con LaTeX. L'autore mostra in che modo si riesca a manipolare e disporre oggetti tridimensionali in una scena con il programma Sketch di Eugene Ressler ed il suo intuitivo linguaggio di *scripting*, ottenendo un'*output* sotto forma di comandi PSTricks o Ti*k*z. Il metodo di lavoro proposto permette di superare le limitazioni che gli utenti di un pacchetto come PSTricks, che pure possiede funzionalità di disegno tridimensionale relativamente avanzate, incontrano ogni qual volta si accingono a voler disegnare e manipolare scene tridimensionali contenenti oggetti non semplici e non *primitivi*.

La didattica di una materia ingegneristica come la Dinamica del Volo è un campo in cui l'autore opera ed al quale si riferiscono gli esempi concreti riportati nell'articolo.

Abstract

This article shows how the combination of LaTeX with the package PSTricks or with Ti*k*z can be used to produce advanced, nice-looking illustrations. As a matter of fact, the creation of drawings representing three-dimensional scenes with scientific or non-trivial annotations is possible with LaTeX. One of the goals of the article is introducing the program Sketch, by Eugene Ressler, and how one can manipulate and put in place objects in a three-dimensional scene by means of its intuitive scripting language. The output of Sketch is a set of PSTricks or Ti*k*z commands that might be included by a master LaTeX document to produce the final picture. The technique proposed here enables to go over the limitations encountered by PSTricks or Ti*k*z users when it comes to representing non-trivial three-dimensional scenes.

As a teacher of engineering subjects related to Flight Dynamics, I have reported some concrete examples that may help to better understand the potential of Sketch and of the workflow proposed in the article.

1 Introduzione

Il titolo dell'articolo non tragga in inganno il lettore: non si vuole esporre un argomento di nicchia. Piuttosto si intende trattare un problema che si presenta nei settori più svariati dell'ingegneria e delle scienze, quello della creazione di buone illustrazioni tridimensionali ad alto contenuto tecnico. L'approccio è quello di presentare dei casi studio basati su esperienze reali, fornendone i particolari implementativi, anziché impostare una discussione che replicherebbe quanto è già trattato nei riferimenti bibliografici.

Introduciamo, per cominciare, il contesto in cui è nato gran parte del materiale di questa comunicazione. Al lettore basti sapere che l'oggetto di studio fondamentale della *Dinamica del Volo* atmosferico è l'evoluzione dei velivoli in un dato sistema di riferimento collegato alla Terra. Insieme al moto dell'aeroplano vengono studiati tutti quei fenomeni fisici che lo determinano. Tra questi ricoprono un'importanza particolare i fenomeni di natura aerodinamica ed ancor più il modo in cui essi dipendono dalla configurazione architettonica del velivolo e dalla condizione di volo. Da qui si comincia a comprendere la necessità in questo campo, e in particolare nella didattica di tali argomenti, di trovare modi efficaci di rappresentare dei concetti fisico-matematici legati al volo. L'esperienza dell'autore è che con l'ausilio di rappresentazioni tridimensionali ben curate l'esposizione di molti concetti si arricchisce in immediatezza visiva e diventa enormemente più efficace. Ciò è vero non solo nella didattica ma anche in comunicazioni di carattere squisitamente scientifico.

La Figura 1 mostra un primo esempio di illustrazione utilizzata tipicamente nello studio della Dinamica del Volo per introdurre alcuni elementi di base. Qui si nota subito come la necessità di definire componenti di vettori – le componenti u, v e w della velocità $\vec{V}$ del baricentro G, quelle p, q ed r della velocità angolare istantanea $\vec{\Omega}$, le componenti X, Y, Z ed $\mathcal{L}$, $\mathcal{M}$, $\mathcal{N}$ delle risultanti delle forze e dei momenti esterni – lungo gli assi di un riferimento standard solidale al velivolo (riferimento degli *assi velivolo* o *body-fixed frame*) richiede naturalmente l'utilizzo di una rappresentazione tridimensionale.

Un secondo esempio di illustrazione è riportato

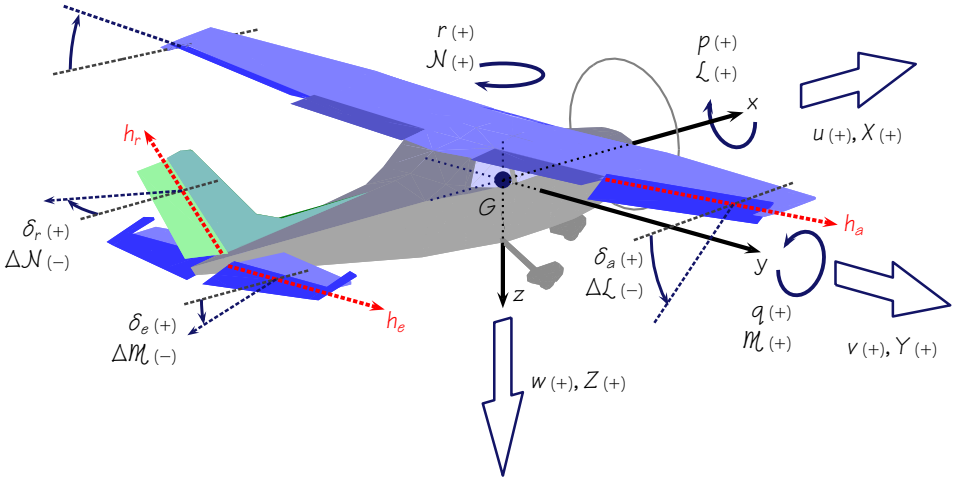


FIGURA 1: Esempio di definizione di alcune tra le principali grandezze fisiche collegate alla dinamica del volo di un velivolo. Per conferire ai disegni l'aspetto di un'illustrazione d'altri tempi l'autore spesso usa una versione modificata del font *Informal Math* della MicroPress Inc. (`http://www.micropress-inc.com/`).

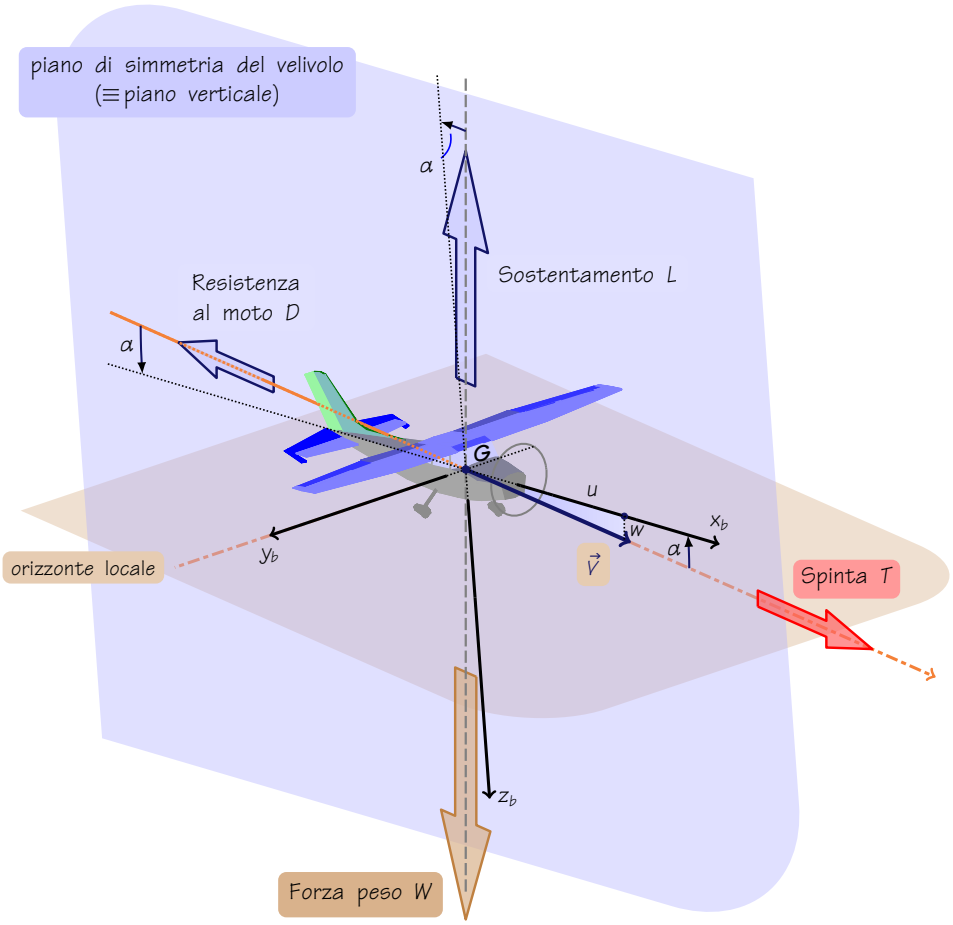
in Figura 2. Da questa si può osservare come, in relazione al fatto che il volo è un fenomeno squisitamente tridimensionale, si rendono spesso necessari dei disegni più o meno articolati se si vogliono rappresentare in maniera efficace alcuni dettagli teoricamente importanti. Quando un velivolo possiede un assetto di volo non simmetrico, Figura 2(b), vi sono cioè un angolo d'attacco α e di derapata β entrambi non nulli, i contributi all'azione aerodinamica totale di alcuni elementi architettonici del velivolo vanno evidenziati mediante l'uso di opportuni pedici (pedici "A"). Ad esempio: impennaggio verticale (*Vertical tail*, "V"), impennaggio orizzontale (*Horizontal tail*, "H"), alettone sinistro/destro (*left/right aileron*, "ail,r/l"), combinazione ala-fusoliera (*Wing-Body*, "WB"). La Dinamica del Volo è un esempio di quei campi in cui l'uso di pedici multipli è molto praticato, al punto che esistono precise convenzioni sull'uso delle notazioni dettate da organismi internazionali. I dettagli a cui l'uso di pedici multipli si riferisce hanno ovviamente una corrispondenza di natura fisica. La Figura 2(b) mostra una soluzione grafica che espone in un solo disegno un concetto che pure è espresso efficacemente da un appropriato modello matematico. Si può scrivere per esempio che:

$$\mathcal{L}_{\mathrm{A}} = \mathcal{L}_{\mathrm{A,WB}} + \mathcal{L}_{\mathrm{A,V}} + \mathcal{L}_{\mathrm{A,ail,r}} + \mathcal{L}_{\mathrm{A,ail,l}} \qquad (1)$$

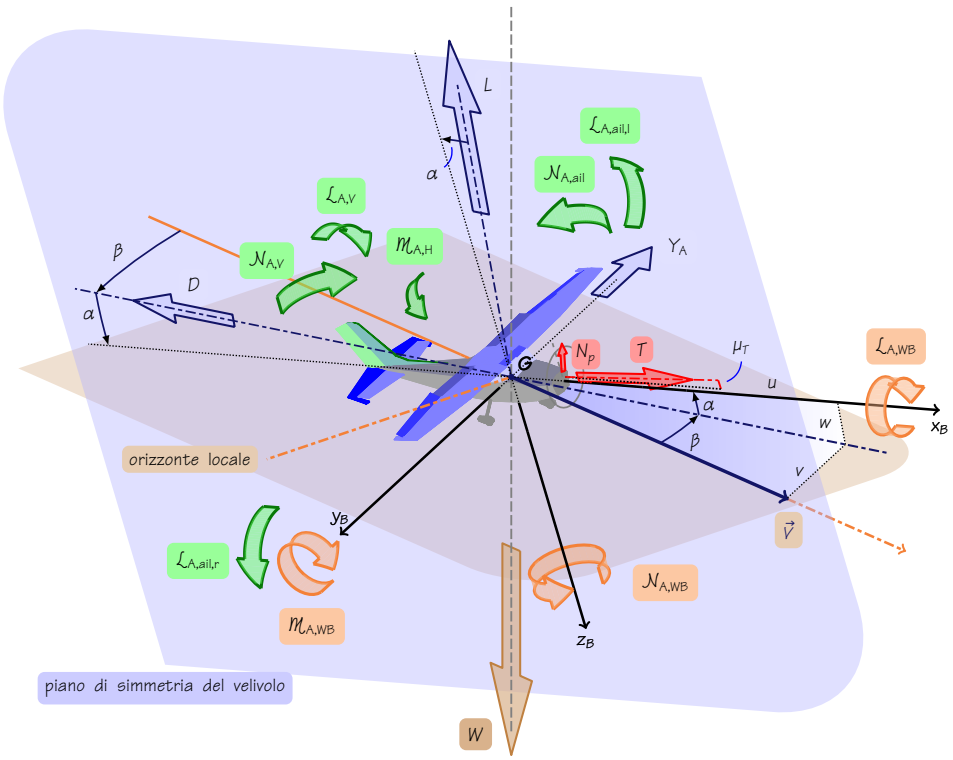
Una formula come quella appena scritta per uno studente di ingegneria aerospaziale dovrebbe essere abbastanza consistente da fargli subito interpretare la quantità $\mathcal{L}_{\mathrm{A}}$ come la *coppia di rollio* di natura aerodinamica (che tende a far ruotare l'aereo intorno all'asse della fusoliera). Gli addendi a secondo membro costituiscono dunque una semplice sovrapposizione di effetti dovuti, rispettivamente, alla combinazione aerodinamica ala-fusoliera ($\mathcal{L}_{\mathrm{A,WB}}$), all'impennaggio verticale di coda ($\mathcal{L}_{\mathrm{A,V}}$) ed ai due alettoni, destro e sinistro ($\mathcal{L}_{\mathrm{A,ail,r}}$ e $\mathcal{L}_{\mathrm{A,ail,l}}$). Si potrebbe osservare a questo punto che la semplicità della formula (1) è superiore in immediatezza all'illustrazione di Figura 2(b). Ciò è ovviamente vero se ci si sofferma al fatto che una grandezza fisica – il primo membro della (1) – è data semplicemente dalla sovrapposizione di più contributi – cioè la somma a secondo membro. La potenza di una illustrazione completa come quella in questione si rivela quando si vuole spingere il fruitore, lo studente o il collega ad esempio, a immaginare da quali altre grandezze fisiche questi addendi dipendono e, per ciascuno di essi, in che misura. Dunque, non basta qui immaginare una semplice somma ma bisogna immaginare ad esempio come cambiano i singoli effetti al variare dell'angolo d'attacco o dell'angolo di derapata. Un tale sforzo cognitivo richiede di immaginare una variazione tridimensionale d'assetto del velivolo. Si vede allora come un'illustrazione tridimensionale ben concepita possa diventare un incredibile ausilio, specialmente per i soggetti non dotati di approccio "visuale" al ragionamento.

Per approfondimenti sulla Dinamica del Volo si rimanda a ETKIN (1982), SCHMIDT (1998), CALCARA (1988), CASAROSA (2004), MENGALI (2001).

L'elemento più rilevante delle figure richiamate sopra è costituito dalla vista assonometrica di un modello di velivolo. Le illustrazioni sono state realizzate con strumenti, per così dire "nativamente" 3D. In altre parole, non si tratta di tradizionali schizzi piani che nascono dalla mano (digitale) di un illustratore, ma di rappresentazioni (proiezioni) in un piano di oggetti posti in una scena tridimensionale. Simili risultati visivi vengono ottenuti tipicamente quando si effettua il *rendering* di una scena in applicazioni di computer grafica come *Blender* (ROOSENDAAL, 2007) o di simili prodotti com-



(a) Condizione di volo simmetrico, orizzontale, ad ali livellate



(b) Condizione di volo non simmetrico, orizzontale, ad ali non livellate

FIGURA 2: Definizione di azioni esterne su di un velivolo. Esse comprendono la forza peso $\vec{W}$ e quelle di natura aerodinamica e propulsiva. In volo simmetrico (in alto) la situazione sembra molto più semplice di quella che si presenta per un velivolo in assetto di volo non simmetrico (in basso), quando vi sono un angolo d'attacco α e di derapata β entrambi non nulli. Nella seconda figura si sono evidenziati con opportuni pedici i contributi all'azione aerodinamica (A) totale di alcuni elementi architettonici del velivolo.

merciali. Un fatto degno di nota è che le illustrazioni ottenute con il metodo proposto in questo articolo, grazie alla tecnologia Postscript e all'uscita di LaTeX in formato PDF, sono intrinsecamente vettoriali.

Nei paragrafi seguenti si cercherà di raccogliere gli elementi che permettano al lettore di creare le proprie illustrazioni 3D e di aggiungervi comodamente il testo ed i simboli matematici desiderati attraverso gli usuali comandi LaTeX.

2 Illustrazioni tridimensionali

Creare illustrazioni tridimensionali di buona qualità da utilizzare in pubblicazioni tecniche e scientifiche non è un compito banale. E ciò si intuisce ovviamente dall'esame delle figure citate in precedenza.

Dal punto di vista editoriale i disegni in questione possiedono delle caratteristiche importanti: (*i*) le annotazioni contengono testo, lettere greche e simboli matematici composti con LaTeX – risulteranno quindi perfettamente integrate con il resto di un eventuale documento composto in LaTeX che le contenga; (*ii*) i disegni sono prodotti in forma vettoriale (formati EPS e PDF); (*iii*) le distanze, le proporzioni e gli angoli sono accurati e visivamente corretti.

2.1 La via di PSTricks

Per la cronaca, nei suoi primi tentativi di creare disegni con simili caratteristiche l'autore ha tentato la via di PSTricks (TUG Boat, 2007; Van Zandt, 2003). PSTricks rappresenta una collezione impressionante di macro per la creazione di disegni ed illustrazioni con TeX/LaTeX. È sufficiente visitare il sito di riferimento di questo pacchetto e consultare la galleria degli esempi per apprezzarne le potenzialità. PSTricks si dimostra uno strumento eccellente per illustrazioni bidimensionali. È possibile anche utilizzare alcune estensioni al pacchetto originale, come ad esempio i pacchetti `pst-3dplot` e `pst-3d`, per creare semplici disegni tridimensionali.

Quando si intende rappresentare una scena tridimensionale non eccessivamente semplice è fondamentale disporre di strumenti pratici che implementino degli operatori matematici di trasformazione geometrica. Tra questi compaiono immancabilmente le matrici di rotazione, le trasformazioni di traslazione, di scalatura e deformazione. Di fatto, questi operatori non sono disponibili in PSTricks oppure sono implementati con funzionalità limitate ad oggetti semplici in estensioni come `pst-3dplot`. PSTricks è un pacchetto basato su TeX e sul linguaggio Postscript e si può pensare di estenderne le possibilità di manipolazione nel mondo 3D. Sebbene sia certamente possibile, ciò è anche estremamente difficile (almeno per i non "TeX guru").

2.2 Il programma Sketch

Esiste fortunatamente uno strumento che permette di ottenere dei risultati avanzati e che al tempo stesso si inserisce agilmente nel flusso di lavoro di produzione delle illustrazioni. Il suo nome è *Sketch*.

Sketch è un programma scritto in linguaggio C da Gene Ressler (Ressler, 2007a), multipiattaforma e distribuito con licenza GPL. Si presenta come uno strumento leggero e semplice per la produzione di disegni 2D e 3D. L'autore di Sketch dichiara nel manuale (Ressler, 2007b) di averlo concepito per creare delle illustrazioni raffinate, che contenessero in prevalenza annotazioni matematiche e che, quando incluse in documenti scientifici, non presentassero dettagli disomogenei all'impostazione tipografica adottata.

I disegni con Sketch sono composti a partire da oggetti primitivi, come punti e linee, da superfici definite per triangolazione ed eventualmente da oggetti testuali. Con opportuni comandi gli oggetti sono posti in una scena e quest'ultima viene proiettata in un piano secondo le classiche regole della prospettiva, in base ad un assegnato punto di vista ed ad una data direzione di osservazione della camera.

Sketch funziona come un comune compilatore ed è programmabile attraverso un linguaggio abbastanza intuitivo. L'output generato può essere del codice PSTricks o Ti*k*z (Tantau, 2006a,b; Fauske, 2007a) da includere opportunamente in un ambiente `figure`. Grazie ad un'apposita opzione della riga di comando è anche possibile ottenere in uscita un sorgente compilabile direttamente da LaTeX e produrre l'illustrazione desiderata in una semplice catena di compilazioni.

L'algoritmo di composizione della scena di Sketch possiede due caratteristiche particolarmente rilevanti: viene applicata per default la rimozione degli oggetti nascosti – meglio nota come algoritmo di *hidden surface removal* – ed esiste la possibilità di includere nei disegni delle annotazioni in forma di comandi LaTeX/PSTricks/Ti*k*z, ancorate in una posizione qualsiasi della scena. Quest'ultima caratteristica offre agli utenti LaTeX degli ovvi vantaggi.

Quando si scopre Sketch ci si rende conto di avere accesso a tutti quegli strumenti matematici e geometrici di cui si ha bisogno per la produzione di disegni tridimensionali e che questi strumenti rendono molto più semplice e veloce il flusso di lavoro. Ad, esempio, in Sketch è così semplice cambiare il punto di vista ed ispezionare la scena che si può dire di avere a disposizione anche uno strumento di *debug visuale* dell'illustrazione.

È opinione dell'autore che i risultati ottenuti con Sketch ripaghino di gran lunga l'utente del tempo speso a studiarne uso e linguaggio.

3 Esempi introduttivi di programmazione in Sketch

Il materiale di questo paragrafo è ispirato in parte al manuale ufficiale di Sketch (Ressler, 2007b) ed in parte al *tutorial* su Sketch presente sul sito web del norvegese Kjell Magne Fauske (Fauske, 2007b,c). Di quest'ultimo sarà istruttivo consultare, in particolare, la pagina dedicata ad una galleria di esempi realizzati con il pacchetto Ti*k*z (Fauske, 2007a).

3.1 Il ciclo di lavoro

Consideriamo per cominciare una semplice scena tridimensionale costituita dalla piramide e dal sistema di assi cartesiani di Figura 3.

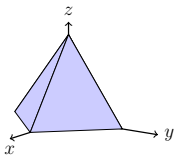


Figura 3: Una piramide con Sketch.

Per comporre questa semplice scena con Sketch basta scrivere un codice sorgente come quello riportato nel listato seguente:

```
def p0 (0,0,0) % "punto", in origine
def vK [0,0,1] % "vettore", asse di rotazione
def dx 2.3     % "scalare"
def dy 2.5
def dz dx

% definiamo un oggetto "drawable"
def Axes {
    % punti sugli assi
    def pX (dx,0,0)
    def pY (0,dy,0)
    def pZ (0,0,dz)
    % uniamo i punti
    line[arrows=->,line width=.8pt](p0)(pX)
    line[arrows=->,line width=.8pt](p0)(pY)
    line[arrows=->,line width=.8pt](p0)(pZ)
    % annotazioni: etichette degli assi
    special |
       \path #1 node[above] {$z$}
             #2 node[below] {$x$}
             #3 node[right] {$y$};|
       (pZ)(pX)(pY)
}

% definiamo un secondo oggetto "drawable"
def Pyramid {
    def p0 (0  ,0,2)
    def p1 (1.5,0,0)
    def n 4
    % definiamo una "trasformazione"
    def tMyRotation rotate(360 / n, (p0), [vK])
    % piramide come rotazione di una linea
    % intorno all'asse [vK] in n=4 passi
    % in gergo: uno "sweep"
    sweep[cull=false,fill=blue!20]
        { n, [[tMyRotation]] } line(p0)(p1)
}

% infine definiamo una scena e
% rendiamo visibile il tutto

% collezione di oggetti "drawable"
def Scene { {Pyramid} {Axes} }

def pEye (7,5,2)    % punto di vista
def pLookAt (0,0,0) % target camera

% disegno della scena
put { view( (pEye), (pLookAt), [0,0,1] ) }
   {Scene}

global { language tikz }
```

Se questo codice viene conservato in un file, ad esempio `pyramid.sk`, e compilato con il comando:[1]

```
> sketch.exe pyramid.sk -o pyramid.tex
```

si otterrà in uscita il file `pyramid.tex`. Date le sue dimensioni contenute, il suo listato viene riportato qui di seguito integralmente:

```
\begin{tikzpicture}[join=round]
\filldraw[fill=blue!20]
   (0,1.897)--(.849,.269)--
   (1.189,-.192)--(0,1.897)--cycle;
\filldraw[fill=blue!20]
   (0,1.897)--(-1.189,.192)--
   (.849,.269)--(0,1.897)--cycle;
\filldraw[fill=blue!20]
   (0,1.897)--(-.849,-.269)--
   (-1.189,.192)--(0,1.897)--cycle;
\draw[arrows=-,line width=.8pt]
   (0,0)--(0,1.897);
\draw[arrows=-,line width=.8pt]
   (0,0)--(-.849,-.269);
\draw[arrows=-,line width=.8pt]
   (0,0)--(1.189,-.192);
\filldraw[fill=blue!20]
   (0,1.897)--(1.189,-.192)--
   (-.849,-.269)--(0,1.897)--cycle;
\draw[arrows=->,line width=.8pt]
   (0,1.897)--(0,2.182);
\draw[arrows=->,line width=.8pt]
   (1.189,-.192)--(1.981,-.321);
\draw[arrows=->,line width=.8pt]
   (-.849,-.269)--(-1.302,-.413);

\path (0,2.182) node[above] {$z$}
      (-1.302,-.413) node[below] {$x$}
      (1.981,-.321) node[right] {$y$};
\end{tikzpicture}
```

Esso contiene dei comandi Ti*k*z, racchiusi in un ambiente `tikzpicture`, necessari a generare il disegno di Figura 3. In questo contesto non interessa entrare nei dettagli dei comandi Ti*k*z, per i quali si rimanda alla documentazione, piuttosto importa chiarire come vada utilizzato Sketch.

Se si predispone il seguente prototipo di file principale di comandi LaTeX:

```
\documentclass[a4paper,12pt]{article}
\usepackage{amsmath}
\usepackage{tikz}
\usetikzlibrary{%
   arrows,snakes,backgrounds,patterns}
\begin{document}
\pagestyle{empty}
\vspace*{\fill}
\begin{center}
%------------------
%%SKETCH_OUTPUT%%
%------------------
```

1. Si supponga di lavorare su piattaforma Windows in una finestra di comandi DOS o, in alternativa, con una *bash shell* di Cygwin, o Linux, o Mac OS X.

```
\end{center}
\vspace*{\fill}
\end{document}
```

e lo si nomina, ad esempio, `main_template.tex`, con il comando

```
> sketch.exe -t main_template.tex \
             pyramid.sk -o main.tex
```

Sketch sostituirà la stringa `%%SKETCH_OUTPUT%%` con i comandi Ti*k*z che generano il disegno e l'utente dovrà semplicemente compilare il file `main.tex` per ottenere l'uscita finale in formato Postscript o PDF:

```
> pdflatex main
> cp main.pdf pyramid.pdf
```

La Figura 4 presenta il flusso di lavoro di un utente che voglia creare un'illustrazione con Sketch, LaTeX ed i pacchetti PSTricks e Ti*k*z.

3.2 Scopriamo la sintassi di Sketch

Dall'esame del listato del file `pyramid.sk` si noteranno molte delle caratteristiche del linguaggio di Sketch. Per brevità si cercherà, ove è possibile, di spiegare il significato delle varie istruzioni con una notazione matematica standard.

Come si intuisce dal nome, il comando `def` viene usato per definire variabili, oggetti della scena, trasformazioni ed entità geometriche in generale. Tra le entità matematiche e geometriche fondamentali messe a disposizione da Sketch troviamo le seguenti. Le grandezze *scalari*, ad esempio:

```
def xA 2.0 def yA 0.0 def zA 0.0 % ← x_A, y_A, z_A
def xB 4.5 def yB 0.0 def zB 2.0 % ← x_B, y_B, z_B
```

I *punti*:

```
def pO (0,0,0)     % ← O ≡ (0,0,0)
def pA (xA,yA,zA)  % ← A ≡ (x_A, y_A, z_A)
def pB (xB,yB,zB)  % ← B ≡ (x_B, y_B, z_B)
def pH ((pB)'x,(pB)'y,0.0) % ← H ≡ (x_B, y_B, 0)
```

I *vettori*:

```
def vK [0,0,1] % ← 0·i + 0·j + 1·k
def vAB [xB-xA,yB-yA,zB-zA]
def vOB (pB)-(pO) % ← v_OB = B − O
def pC (pA)+[0,2,0]  % ← C = A + 2j
def vAC (pC)-(pA) % ← v_AC = C − A
% prodotto vettoriale
def vN_ABC [vAB]*[vAC] % ← v_AB × v_AC
def vUnitN_ABC unit([vN_ABC]) % ← versore n_ABC
% proiezione sul piano xy
def vNxy [ [vUnitN_ABC]'x,
           [vUnitN_ABC]'y,
           0 ]             % ← n_xy · k ≡ 0
% centro del triangolo
def pM0 ( ((pA)'x+(pB)'x+(pC)'x)/3,
          ((pA)'y+(pB)'y+(pC)'y)/3,
          ((pA)'z+(pB)'z+(pC)'z)/3 )
% punto lungo la normale
def pM1 (pM0)
        +2.0*[vUnitN_ABC] % ← M_1 ≡ M_0 + 2n_ABC
```

Le *trasformazioni*:

```
% rotazione
def tRot rotate(180,(pH),[0,0,1])
% traslazione
def tTransl translate(2*[1,0,0])
% scaling
def sfz 0.8 def tScaleZ scale([1.0,1.0,sfz])
% roto-traslazione
def tMyT [[tRot]] then [[tTransl]]
```

Le istruzioni riportate sopra costituiranno un secondo esempio di composizione di una scena. Esse lasciano intuire la enorme potenzialità del simbolismo messo a disposizione dal linguaggio Sketch. Ciò che si vuole ottenere è il disegno di Figura 5.

Per quanto riguarda la designazione dei nomi delle variabili in Sketch, valgono le stesse regole del linguaggio C. Lo stesso dicasi per le usuali operazioni aritmetiche tra grandezze scalari. Esiste anche la possibilità di usare le più comuni funzioni matematiche. Ad esempio:

```
% distanze tra punti, moduli di vettori
def dAH |(pH)-(pA)| % ← d_AH = |H − A|
def dBH |(pB)-(pH)|
% funzioni trigonometriche
def alpha atan2(dBH,dAH) % gradi
def dAB |dAH/cos(alpha)| % valore assoluto
                         % cos() accetta gradi
```

Per quanto riguarda i punti dello spazio di coordinate (x, y, z), questi saranno definiti con la notazione usuale che fa uso di parentesi tonde (vedi definizione di `pO`, `pA`, ecc.). I nomi di variabili di tipo punto saranno utilizzabili in successive operazioni racchiudendoli tra parentesi tonde (vedi l'uso di `(pO)`, `(pA)`, `(pB)`, nelle definizioni di `pH`, `vOB`, `pC` ecc.).

Per quanto riguarda i vettori, intesi come *direzioni* nello spazio, la loro definizione in Sketch sarà simile a quella dei punti ma farà uso delle parentesi quadre (vedi definizioni di `vK` e `vAB`). Un vettore potrà anche essere definito come *differenza tra due punti* (vedi definizioni di `vOB`, `vAC`, ecc.) ed, analogamente, un punto potrà essere definito come somma di un altro punto e di un vettore, come se quest'ultimo fosse un vettore *applicato* (vedi definizioni di `pC` e `pM1`). I nomi di variabili di tipo vettore saranno utilizzabili in successive operazioni racchiudendoli tra parentesi quadre (vedi `[vAB]`, `[vAC]` nella definizione di `vN_ABC`). Se in una data espressione una variabile vettore viene manipolata utilizzando le parentesi tonde anziché quadre, ad esempio `(vAB)`, Sketch segnalerà un errore. Analoga situazione si avrà per uno scorretto uso delle parentesi con le variabili punto.

Sia per le variabili di tipo punto che per quelle di tipo vettore esiste la possibilità di estrarne le componenti con gli operatori `'x`, `'y`, `'z` (vedi le definizioni di `pH`, `vNxy` e `pM0`).

Per quanto riguarda le trasformazioni, esse sono in generale costituite da rotazioni, traslazioni, scalature, e da loro combinazioni. Nel linguaggio di Sketch queste trasformazioni vengono definite con una sintassi molto intuitiva. Ad esempio, la definizione di `tRot` lascia intendere che si tratta di una rotazione – in notazione matematica: $\mathcal{T}_{\mathrm{rot},H,z}$ – di 180° intorno ad un asse passante per il punto `(pH)` e parallelo al versore dell'asse z, `[0,0,1]`. Un

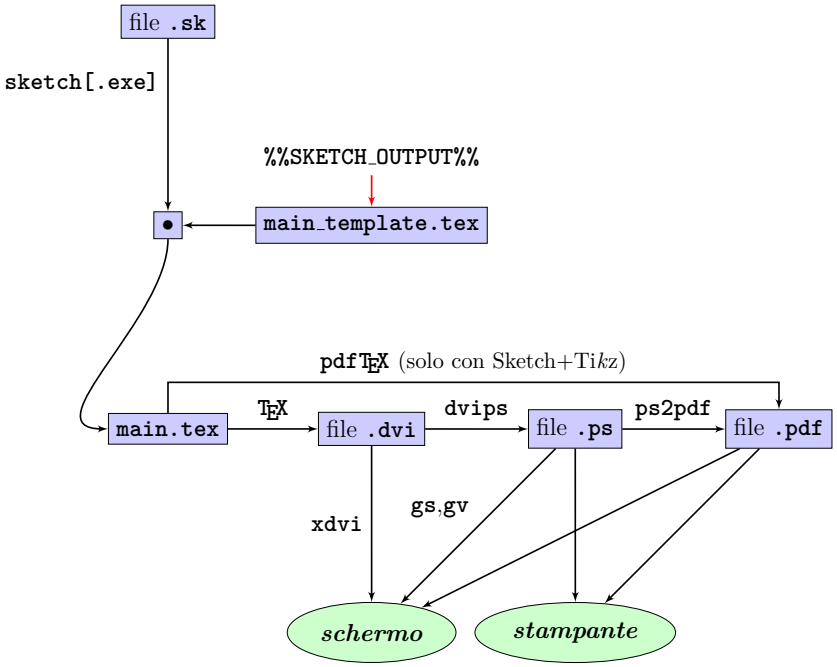


Figura 4: Il flusso di lavoro con Sketch e LaTeX.

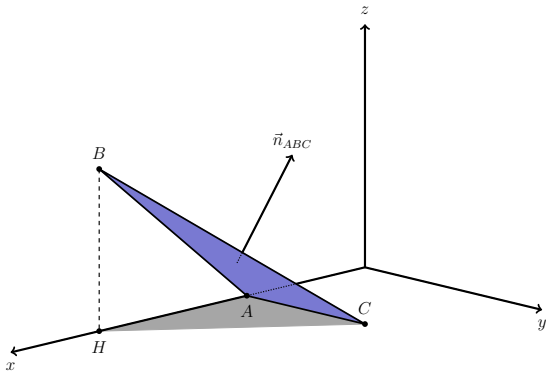


Figura 5: Una scena composta con Sketch.

punto P_2 ottenuto dal punto P_1 attraverso questa trasformazione sarà dato dall'istruzione:

```
def pP2 [[tRot]]*(pP1) % ← P2 = T_rot,H,z(P1)
```

Come si vede da questo esempio, una volta definite, le trasformazioni esse vengono riutilizzate racchiudendone il nome tra doppie parentesi quadre: `[[tRot]]`. L'esempio mostra anche che la trasformazione di un punto si ottiene attraverso un'opportuna operazione di moltiplicazione. Ciò corrisponde al fatto che Sketch definisce una trasformazione come $\mathcal{T}_{\mathrm{rot},H,z}$ attraverso un operatore matriciale di uno spazio di *coordinate omogenee*.

Le coordinate omogenee di un punto $P \equiv (x, y, z)$ dello spazio euclideo tridimensionale sono date da una qualsiasi quaterna (ξ, η, ζ, w) di numeri reali tali che: $w \neq 0$, $x = \xi/w$, $y = \eta/w$ e $z = \zeta/w$. Dunque, il punto espresso in coordinate (ξ, η, ζ, w) nello spazio ampliato è equivalente al punto reale $(\xi/w, \eta/w, \zeta/w)$. I punti in coordinate omogenee con coordinata w nulla sono detti impropri, e non hanno nessun significato geometrico nello spazio cartesiano, ma possono rappresentare un punto all'infinito, nella direzione del vettore tridimensionale (x, y, z). Le coordinate omogenee permettono dunque di rappresentare punti all'infinito, e consentono di esprimere *tutte* le trasformazioni di coordinate in forma matriciale. L'insieme costituito da tutte le quaterne non nulle forma uno spazio proiettivo tridimensionale.

Le coordinate omogenee sono particolarmente vantaggiose in computer grafica per il fatto non banale che qualunque trasformazione affine è rappresentabile con un prodotto tra matrici, ma lo è anche la stessa proiezione prospettica. L'uso di coordinate omogenee è particolarmente importante, perché implica che Sketch è in grado di operare trasformazioni per proiezioni in prospettiva.

Le rimanenti trasformazioni sono anch'esse definite intuitivamente, vedi ad esempio la definizione della traslazione `tTransl` di entità `2`, parallelamente all'asse x – in notazione matematica, una funzione $\mathcal{T}_{\mathrm{trasl},x}$; o anche la scalatura, una riduzione dell'80%, della coordinata z definita come `tScaleZ`. Esiste inoltre la possibilità di definire delle trasformazioni composte, cioè di definire delle trasformazioni come sequenza di altre, come nel caso della trasformazione di roto-traslazione definita dalla variabile `tMyT`. La sequenza di trasformazioni viene ottenuta attraverso l'uso della parola chiave `then`.

Accanto agli oggetti definiti in una scena, Sketch offre la possibilità di inserire annotazioni costituite da vere e proprie finestre di comandi LaTeX. Ciò è ottenuto mediante l'uso del comando `special`. Per esempio, un utente di PSTricks po-

trebbe pensare di utilizzare il comando `\pscircle` per ottenere un punto pieno di diametro `2pt` in corrispondenza delle posizioni dei punti A, B e C nella scena e contemporaneamente posizionare le etichette `$A$`, `$B$` e `$C$` con il comando `\uput`:

```
special |
    \pscircle*[fillcolor=black]#1{2pt}
    \uput{2pt}[-90]{0}#1{$A$}
    \pscircle*[fillcolor=black]#2{2pt}
    \uput{2pt}[  0]{0}#2{$B$}
    \pscircle*[fillcolor=black]#3{2pt}
    \uput{2pt}[ 90]{0}#3{$C$}
|(pA)(pB)(pC)
```

Il comando `special` si comporta come una funzione che accetta un numero variabile di parametri di tipo punto. I parametri in questo caso sono i punti `(pA)`, `(pB)` e `(pC)`. Le loro coordinate tridimensionali riferite alla scena verranno trasformate da Sketch in coordinate piane secondo regole prospettiche e sostituite nel testo del comando `special` racchiuso dai delimitatori `|...|` al posto delle stringhe `#1`, `#2` e `#3`. Ciò consente di ottenere delle annotazioni e in generale degli effetti grafici di basso livello in posizioni precise di un disegno, corrispondenti a punti dello spazio a tre dimensioni. L'utente non è tenuto a conoscere la posizione finale sul piano del foglio delle variabili punto che intende passare al comando. Questa cambierà infatti se verrà variata la posizione del punto di osservazione o la direzione di osservazione della camera (vedi comando `view` nel listato del file `pyramid.sk` o più avanti).

Il comando `special` è un comando molto flessibile e permette di ottenere effetti grafici non banali, combinando la potenza degli ambienti e dei pacchetti LaTeX e la possibilità di manipolare accuratamente punti della scena tridimensionale. Un esempio è costituito dal disegno della curva tridimensionale di Figura 13. Per via del fatto che la versione attuale di Sketch non permette di disegnare delle curve spaziali con un comando apposito, che accetti dei punti come argomenti, la traiettoria del velivolo è stata ottenuta in maniera artificiosa. Si è definito un insieme di punti lungo la traiettoria, nella scena tridimensionale. Tali punti sono stati poi passati come argomenti ad un comando `special` all'interno del quale si è provveduto, con comadi PSTricks, a disegnare una curva piana che unisse le proiezioni prospettiche nel piano del foglio dei punti della curva tridimensionale. Il risultato grafico, seppure approssimato, appare soddisfacente.

3.3 Gerarchie di oggetti e trasformazioni

Vediamo ora come si applica una trasformazione come `tMyT` agli oggetti della scena di Figura 5. Il risultato che vogliamo ottenere è un disegno come quello di Figura 6.

Per arrivare ad un uso elegante delle trasformazioni, vediamo come si può organizzare una gerarchia di oggetti grafici e di annotazioni nel disegno di Figura 5. Introduciamo il concetto di *polygon*. Nel disegno di partenza compaiono due superfici piane triangolari, una di vertici (A, B, C), l'altra di vertici (A, H, C). In Sketch, con uscita Ti*k*z, questi due triangoli sono definiti come segue:

```
def Triangle1 {
    % solo contorni, no riempimento
    def stylecontour
       [lay=over,line width=1.4pt,fill=none]
    polygon[stylecontour](pA)(pB)(pC)
    % riempimento colorato
    def stylefill
       [style=solid,
          color=blue!50!gray!70]
    polygon[stylefill](pA)(pB)(pC)
    % cerchi pieni per ogni punto
    special |
        \fill [style=solid] #1 circle (2pt);
        \fill [style=solid] #2 circle (2pt);
        \fill [style=solid] #3 circle (2pt);
    |(pA)(pB)(pC)
}
def Triangle2 {
    % riempimento, no contorni
    def stylefill
       [line width=0pt,style=solid,
          color=gray!70]
    polygon[stylefill](pA)(pH)(pC)
    special |
        \fill [style=solid] #1 circle (2pt);
    |(pH)
    % tratteggio BH
    special|
        \draw[-,dashed,line width=0.8pt]
            #1 -- #2;
    |(pB)(pH)
}
```

Le istruzioni precedenti definiscono i due oggetti grafici `Triangle1` e `Triangle2` contenenti il comando `polygon`. Quest'ultimo accetta delle opzioni tra parentesi quadre per il controllo del riempimento e dei contorni. Nel caso particolare le opzioni seguono la sintassi di Ti*k*z, sono definite come stringhe di caratteri (vedi definizioni di `stylecontour` e `stylefill`) e passate infine al comando racchiuse da parentesi quadre (vedi ad esempio `polygon[stylefill](pA)(pB)(pC)`). Il comando `polygon` definisce in questo caso una superficie triangolare piana che unisce i tre punti passati come argomenti. Si rimanda al manuale di riferimento per approfondimenti sull'uso di questo comando.

I due triangoli rappresentano un esempio di oggetti "drawable", nel gergo di Sketch, cioè di oggetti *disegnabili*. Al contrario, gli oggetti punto come `(pA)`, `(pB)`, e così via, gli oggetti vettore e le trasformazioni definite sopra *non* sono entità disegnabili. Per il meccanismo di composizione della scena e di produzione del disegno finale sarà necessario che i due triangoli, dopo la loro definizione, siano *posti nella scena* con un'apposita sintassi. Il disegno di oggetti *drawable* sarà ottenuto racchiudendone il nome tra parentesi graffe. Ad esempio, l'istruzione:

```
{Triangle1} {Triangle2}
```

permette di ottenere il disegno dei triangoli per

valori di default della posizione del punto di vista e dell'orientamento della camera. L'utente potrà intervenire tanto sulla posizione dell'osservatore della scena, tanto sulla direzione di osservazione con una trasformazione apposita, `view`, in combinazione con il comando `put`. Si potrà avere ad esempio (vedi anche il listato di `pyramid.sk`):

```
def pEye (2,2,0.7)  % punto di vista
def pLookAt (0,0,0) % target camera
% disegno
put { view( (pEye), (pLookAt), [0,0,1] ) } {
   {Triangle1} {Triangle2}
}
```

Il comando `put` serve a produrre il disegno di oggetti *drawable* dopo avervi applicato una o più trasformazioni. Nell'esempio la trasformazione utilizzata è `view`, che accetta tre argomenti: un punto, `(pEye)`, che identifica la posizione dell'obiettivo della camera, un secondo punto, `(pLookAt)`, che identifica la posizione nella scena *puntata* dalla camera, ed un vettore, nell'esempio `[0,0,1]`, che indica la direzione verticale (si consulti il manuale di Sketch per i dettagli). Due possibili alternative di uso di `put` in questo contesto sono date dagli esempi seguenti:

```
put { view( (pEye), (pLookAt), [0,0,1] )
      then [[ScaleZ]] } {
   {Triangle1} {Triangle2}
}
```

oppure

```
put { view( (pEye), (pLookAt), [0,0,1] ) {
   put { translate([0,0,-3]) } {
       {Triangle1}
   }
   {Triangle2}
}
```

in cui si vede che `put` accetta anche sequenze di trasformazioni e che comandi `put` possono essere innestati l'uno nell'altro per un maggiore controllo del posizionamento degli oggetti nella scena.

Sketch permette di costruire degli oggetti *drawable* come delle gerarchie, per composizione di altri oggetti *drawable*. Se si intende disegnare sempre insieme i due triangoli `Triangle1` e `Triangle2` basta definire un nuovo oggetto come segue:

```
def Triangles {
   {Triangle1} {Triangle2}
   {VecNormal2Triangle1}  % vettore normale
}
```

È interessante vedere a questo punto quali sono gli effetti grafici di una trasformazione come `tMyT` definita in precedenza. Il risultato della rototraslazione applicata ai triangoli ed al vettore definiti sopra come unico oggetto *drawable* `Triangles` è mostrato in Figura 6.

Il disegno della nuova coppia di triangoli si otterrà semplicemente con l'istruzione seguente:

```
% nuovo drawable
def NewTriangles [[tMyT]]*{Triangles}
% disegno
put { view( (pEye), (pLookAt), [0,0,1] ) {
   {NewTriangles}
}
```

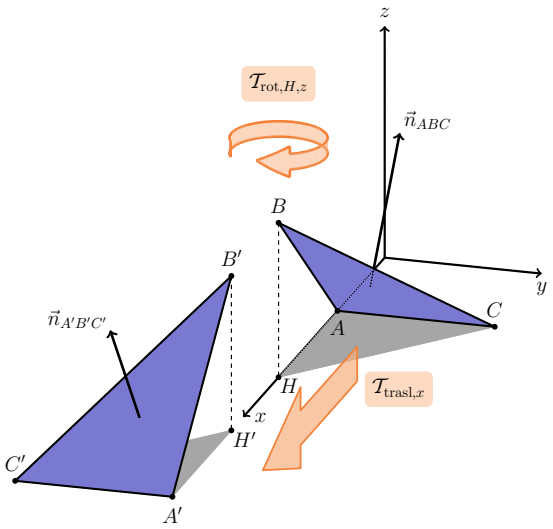


FIGURA 6: Una trasformazione che ruota e trasla i triangoli di Figura 5. Il punto di vista è stato cambiato (vedi sopra).

Un disegno completo come quello di Figura 6 sarà prodotto dalle istruzioni:

```
% Scena: collezione di oggetti drawable
def Scene {
   {Triangles} {Labels}
   {NewTriangles} {LabelsNew}
}
% disegno della scena
put { view( (pEye), (pLookAt), [0,0,1] ) }
   {Scene}
% produci output per Tikz
global { language tikz }
```

3.4 Operazioni di disegno cicliche

In Sketch esiste la possibilità di effettuare delle operazioni di disegno ripetute. Le funzionalità di disegno cicliche di Sketch, sebbene attualmente limitate, si presentano comunque come degli strumenti utili e snelli per particolari esigenze di disegno. Il concetto che ne sta alla base è quello di poter disegnare un certo numero di istanze di un oggetto *drawable*, ciascuna delle quali è ottenuta dalla precedente attraverso una trasformazione prefissata. Il comando corrispondente è dato dalla parola chiave `repeat` e si rimanda il lettore al manuale di riferimento per un approfondimento dei dettagli.

Un comando simile a `repeat`, ma che permette di creare curve e superfici a partire da oggetti trasformati ripetutamente nello spazio, è dato dal comando `sweep`. La Figura 7 presenta il disegno di una curva dello spazio detta *elica*. Essa si può ottenere componendo un moto di rotazione a velocità angolare costante del punto H_0 intorno all'asse z con un moto di traslazione a velocità costante parallelo all'asse z. Il risultato è una curva la cui tangente ha una pendenza θ costante rispetto al piano xy. La tangente della pendenza è detta *passo* e viene tipicamente indicata con $p = \tan\theta$.

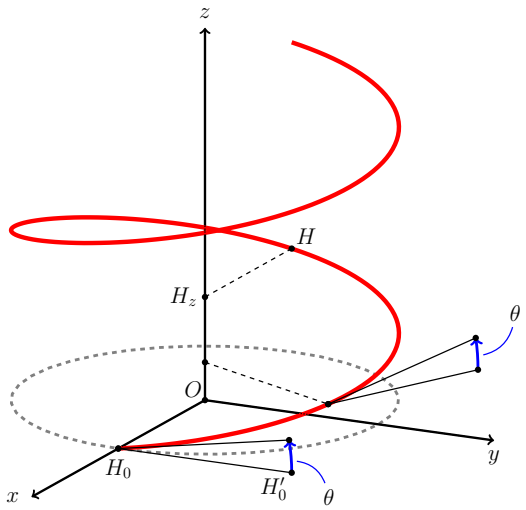


FIGURA 7: Un'elica di asse z e passo $p = \tan\theta$.

La curva di Figura 7 si otterrà con i comandi seguenti:

```
def pO (0,0,0) def pH0 (3,0,0)
def vK [0,0,1]
def Helix {
    % parametri
    def nsteps 180
    def angle_step (360+180)/nsteps
    def pitch_angle 10
    def radius |(pH0)-(pO)|
    def pitch radius * (angle_step/57.3)
        * sin(pitch_angle)/cos(pitch_angle)
    def tHelix rotate(angle_step,(0,0,0),[vK])
               then translate(pitch*[vK])
    % elica
    sweep[line width=2.8pt, color=red] {
        nsteps, [[tHelix]]
    } (pH0)
    % punto pieno in H0
    special |
       \fill [style=solid] #1 circle (2pt);
    |(pH0)
    % cerchio di base, z=0
    def BaseCircle {
        sweep[line width=1.8pt, color=gray,
                style=dashed] {
           36,
           rotate(360/36, (0,0,0), [vK])
        } (pH0)

    }
    % commenta per non disegnare il cerchio
    {BaseCircle}
}
put { view((pEye), (pLookAt), [0,0,1]) } {
    {Helix}
}
```

L'istruzione `sweep` viene utilizzata sia per disegnare l'elica che per disegnare la circonferenza tratteggiata sul piano $z = 0$. Per usare questo comando bisogna specificare: (*i*) il numero di cicli di disegno (vedi `nsteps`), (*ii*) la trasformazione da applicare all'istanza precedente per ottenerne una corrente (vedi `[[tHelix]]`) e (*iii*) un oggetto *generatore*, nel nostro esempio un punto (vedi `(pH0)` nella definizione dell'elica e di `BaseCircle`).

Si osservi come il comando `sweep` sia stato utilizzato nel listato di `pyramid.sk` specificando una linea (vedi `line(p0)(p1)`) come oggetto generatore ed ottenendo come risultato la superficie laterale della piramide di Figura 3.

Un'ulteriore esempio di effetto grafico ottenuto per mezzo di un ciclo è costituito dalle annotazioni di Figura 7 che mostrano l'entità dell'angolo θ. Un'istruzione di `sweep` viene qui utilizzata per disegnare gli archi di circonferenza che terminano con una freccia. Se si considera, ad esempio, il punto H_0, per applicare correttamente il comando `sweep` va definito un punto H_0' lungo la retta orizzontale uscente da H_0. Il punto H_0' diventa il punto generatore dello *sweep* e viene trasformato nel punto H_0'', appartenente alla tangente all'elica passante per H_0. In linguaggio Sketch si scriverà:

```
% vettore lungo il raggio
def vR0 unit( (pH0)-(pO) )
% vettore orizzontale
def vH0 unit( [vR0]*[vK] )
% punto H0'
def pH01 (pH0)+3*[vH0]
def tH02 rotate(pitch_angle,(pH01),[vR0])
% punto H0'' lungo la tangente all'elica
def pH02 [[tH02]]*(pH01)
% arco
sweep[line width=1.8pt, color=blue,
      style=solid,arrows=->]{
     4, rotate(pitch_angle/4,(pH0),[vR0])
}(pH01)
% linee e punti con Tikz
special|
   \draw[-,solid,line width=0.8pt]#1 -- #2;
   \fill [style=solid] #1 circle (2pt);
   \fill [style=solid] #2 circle (2pt);
   \fill [style=solid] #3 circle (2pt);
   \draw[-,solid,line width=0.8pt]#1 -- #3;
|(pH0)(pH01)(pH02)
% simbolo θ con Tikz
special|
   \path[line width=0pt]#1
      to node [node distance=0pt,pos=0.4,
         pin={[pin distance=24pt,
            pin edge={bend left,blue,thick}]
               -45:$\theta$}]{} #2;
|(pH01)(pH02)
```

Per un punto H qualsiasi lungo l'elica, l'effetto voluto viene realizzato potendo comodamente ottenere in Sketch un vettore dal prodotto vettoriale di altri due. Si costruiranno: un punto H' staccandolo lungo una retta orizzontale uscente da H e normale al raggio locale, ed un punto H_z proiettando H sull'asse. Osservando che i due vettori $H - H_z$ ed $(H' - H) \times \vec{k}$ sono paralleli, si potrà definire un vettore da utilizzare nell'istruzione `sweep` come asse di rotazione. Tale operazione avrà come centro il punto H e come punto generatore il punto H'. L'effetto grafico voluto è mostrato in Figura 7.

4 Uso avanzato di Sketch

Le caratteristiche del linguaggio di composizione di una scena messe a disposizione da Sketch sono tali da poter dire che le illustrazioni tridimensionali con esso ottenibili sono di fatto delle illustrazioni avanzate.

La possibilità di costruire delle superfici dello spazio come reticolati, cioè come insiemi di tasselli

triangolari, attraverso il comando `polygon`, l'algoritmo di *hidden surface removal* (anche noto come algoritmo "del pittore") e la possibilità di definire trasformazioni di coordinate aprono la strada ad un uso ancora più spinto di Sketch.

Un utente potrebbe pensare, ad esempio, di costruire una collezione di modelli 3D e di utilizzarli a proprio uso e consumo. Sarà possibile così comporre delle scene avanzate inserendovi uno o più oggetti opportunamente disposti e corredati di specifiche annotazioni, come avviene in molte illustrazioni tecniche. Alla luce di ciò, si pensi ai popolari formati grafici STL o OBJ, che descrivono superfici 3D come tassellazioni, e a tutti quei modelli tridimensionali liberamente scaricabili da internet o creabili con programmi *open source* come Blender. Con queste idee in mente l'autore ha creato col tempo, a partire da modelli in formato STL, una libreria personale di modelli di velivolo in formato Sketch. Per fare ciò è bastato convertire le tassellazioni STL in istruzioni `polygon` mediante l'uso di un'apposito programma (De Marco, 2007) basato sul software *open source* `IVCon-TL` di John Burkardt (Burkardt, 2007).

A seconda dell'esigenza specifica, un dato modello di velivolo può essere opportunamente composto e posizionato nella scena, una o più volte. Si veda a tal proposito la Figura 8.

Il semplice modello di aereo di Figura 8 è stato concepito per scopi didattici. Si distingue la diversa colorazione di alcune parti architettoniche: la fusoliera, il tettuccio, le ali (dorso e ventre), la deriva verticale, gli alettoni, la parte fissa e la parte mobile (equilibratore) del piano di coda orizzontale, il timone. Ciascuno di questi sottocomponenti risiede in un file separato, contenente gli appositi comandi `polygon` e viene richiamato all'occorrenza con il comando `input`. Ad esempio:

```
% polygon files
def Aircraft_Fuselage {
    def bodystyle [line width=0pt,
        style=solid,color=gray!70]
    input{small_aircraft_fus.sk}
}
def Aircraft_Canopy {
    def bodystyle [line width=0pt,
        style=solid,color=mylightblue!100!gray]
    input{small_aircraft_canopy.sk}
}
def Aircraft_Fin {
    def bodystyle [line width=0pt,
        style=solid,color=mydarkblue!40!gray]
    input{small_aircraft_fin.sk}
}
% ...
% The Aircraft
def Aircraft {
    % ...
    {Aircraft_Fuselage} {Aircraft_Canopy}
    {Aircraft_Fin} {Aircraft_HTail}
    {Aircraft_WingsTop} {Aircraft_WingsBottom}
    %----------------------------------------
    % Scommentare ed editare come si conviene
    % se si vogliono ruotare questi oggetti
    %----------------------------------------
    % put{
```

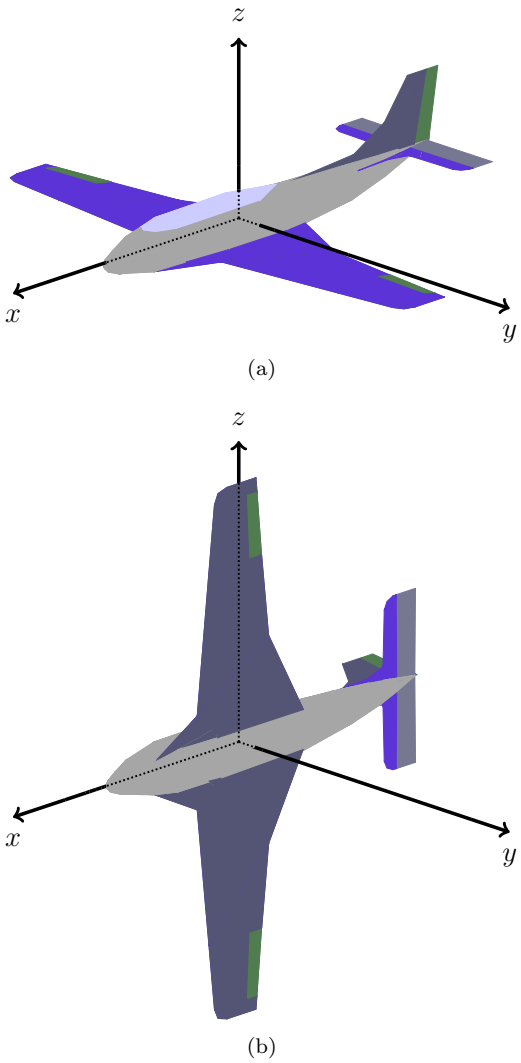


Figura 8: Un semplice modello tridimensionale di velivolo con Sketch.

```
    %   rotate( 0, (pElevatorHingeB),
    %     [vElevatorAxis] ) }
        {Aircraft_Elevator}
    % put{
    %   rotate( -30, (pRudderHingeA),
    %     [vRudderAxis] ) }{Rudder}
        {Aircraft_Rudder}
    % put{ rotate( -30, (pAileronHingeRA),
    %     [vAileronAxisR] ) }{AileronR}
        {Aircraft_AileronR}
    % put{ rotate( -30, (pAileronHingeLA),
    %     [vAileronAxisL] ) }{AileronL}
        {Aircraft_AileronL}
}
```

Le definizioni riportate sopra si fondano sull'istruzione `input` e sulla convenzione che ciascun file contenente la descrizione di una superficie presenti i comandi `polygon` con un campo delle opzioni fittizio del tipo `[bodystyle]`. Un esempio di file `small_aircraft_elevator.sk` è il seguente:

```
%-----------------------------------------
% SKETCH file: small_aircraft_fus.sk
% Original data: small_aircraft_fus.stl
polygon[bodystyle]
   ( 9.050886, -624.006042, 41.119484 )
   ( -0.105902, -693.260681, 43.389565 )
   ( 240.739075, -695.190979, 43.000031 )
```

```
polygon[bodystyle]
   ( 9.050886, -624.006042, 41.119484 )
   ( 240.739075, -695.190979, 43.000031 )
   ( 240.739075, -624.028137, 43.000027 )
% ...
%----------------------------------------EOF
```

Il comando `input`, come in altri linguaggi, permette di includere comandi contenuti in un file separato. Nell'esempio si è usata la possibilità di definire in Sketch delle variabili stringa di caratteri. La stringa `bodystyle` viene definita localmente a ciascun oggetto (vedi ad esempio `Aircraft_Fuselage`) prima di richiamare i comandi del tipo

```
polygon[bodystyle](<p1>),(<p2>),(<p3>)
```

contenuti nel file esterno (vedi ad esempio il frammento di file `small_aircraft_fus.sk` riportato sopra). La variabile `bodystyle` ha *campo di visibilità locale*, cioè non risulra definita al di fuori della definizione di un oggetto come `Aircraft_Fuselage`. Ciò permette di ridefinire tale variabile più volte, una per ciascun oggetto che andrà a comporre l'oggetto *drawable* gerarchicamente più importante: `Aircraft`. Sarà quest'ultimo ad essere disegnato nella scena con un opportuno comando `put`. Ad esempio, il disegno del velivolo di Figura 8(b) si otterrà con comandi del tipo:

```
put { view((pEye), (pLookAt), [0,0,1]) } {
    put { rotate(90,(0,0,0),[1,0,0]) }{
        {Aircraft}
    }
}
```

5 Piccola galleria di illustrazioni

Per terminare con degli esempi, si presenta una piccola galleria di illustrazioni realizzate con Sketch e sfruttando le caratteristiche sia di Ti*k*z che di PSTricks.

L'illustrazione di Figura 10 è stata creata per mostrare le azioni esterne su un velivolo durante una virata. Si noti l'uso di trasparenze per mezzo di comandi Ti*k*z e l'uso di frecce tridimensionali sia dritte che curve. Queste ultime sono state definite analiticamente in Sketch e vengono utilizzate con opportune impostazioni di colore e di fattore di scala. Nell'esempio specifico le frecce aiutano a capire che la portanza L e la resistenza D aerodinamiche del velivolo sono disposte in un piano inclinato di ϕ_W sulla verticale locale.

L'illustrazione di Figura 11 è stata creata per mostrare un caso particolare di evoluzione, l'evoluzione di *pull-up* sostenuta fino a compiere un cosiddetto *loop*, durante la quale le storie temporali degli angoli di Eulero del velivolo presentano delle discontinuità. Le annotazioni e le colorazioni sono state ottenute con PSTricks.

I disegni di Figura 12 sono stati anch'essi ottenuti come quello precedente e rappresentano classiche illustrazioni in cui l'orientamento del velivolo nello spazio, definito dagli angoli di Eulero, è visualizzato per mezzo di una sospensione cardanica. Gli oggetti grafici sono stati modellati con un software CAD, esportati in formato STL e convertiti in poligoni in formato Sketch.

L'illustrazione di Figura 13 mostra un secondo esempio di evoluzione di volo. Come per la Figura 11, il modello 3D del velivolo è quello presentato in Figura 8. In questo esempio uno solo degli angoli di Eulero, l'angolo detto di rollio ϕ, presenta una discontinuità nel tempo.

6 Vantaggi e svantaggi nell'uso di Sketch

6.1 Osservazioni

L'uso di Sketch presenta vantaggi e svantaggi. Questi saranno più o meno sentiti a seconda del tipo di utente, del suo *background* culturale, della sua esperienza. Ad esempio alcuni potrebbero muovere la seguente osservazione: *Quale vantaggio comporta l'uso di Sketch rispetto all'ovvia alternativa di poter realizzare con uno strumento di modellazione tridimensionale delle viste prospettiche in formato bitmap o vettoriale? Queste immagini potrebbero essere importate ed annotate con codice LATEX extra che consenta di collocare le scritte nei punti giusti* (vedi Figura 9). *Oppure, se le immagini sono vettoriali, si potrebbero inglobare in esse dei frammenti di codice ad hoc da poter manipolare con LATEX in fase di produzione del documento finale.*

Una simile osservazione vale per chi propone l'uso di PSTricks e Ti*k*z: *Perché ci si dovrebbe ostinare ad usare simili pacchetti quando esistono applicazioni commerciali e non*[2] *con le quali poter produrre illustrazioni anche molto complesse?* È una questione di "look and feel", direbbero gli americani. Eppure, il sito web di PSTricks è uno dei più visitati e dei più attivi; lo si vede dall'evoluzione che hanno avuto negli ultimi due anni le sezioni dedicate agli esempi ed ai pacchetti collegati.

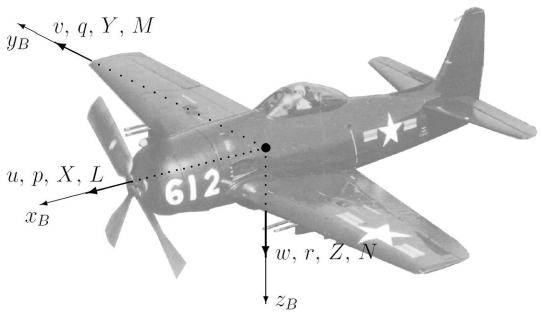


Figura 9: Esempio di immagine bitmap con annotazioni apposte nei punti giusti.

2. Ad esempio, Macromedia FreeHand, Adobe Illustrator, Corel Draw, Inkscape, XFig, WinFig.

Una risposta ragionevole alle osservazioni precedenti è che ci sono diverse tipologie di utenti – a volte completamente opposte – e di volta in volta il metodo di lavoro più appropriato dipende anche dal carattere e dalla complessità dell'illustrazione che si intende realizzare. Non trascurabile altresì è lo scopo al quale l'illustrazione è destinata: una pubblicazione, una monografia, una dispensa didattica informale, una presentazione, un rapporto tecnico.

Per valutare le possibili situazioni di vantaggio derivanti dall'uso di Sketch va tenuto presente che tale approccio è destinato ad utilizzatori di LaTeX/PSTricks/Ti*k*z, cioè ad utenti che hanno sposato ed in alcuni casi si sono piegati ad un metodo di lavoro non-WYSIWYG. In ogni caso la produzione di un'illustrazione di buona qualità si concluderà al termine di un processo iterativo durante il quale saranno probabilmente necessarie diverse ricompilazioni prima di raggiungere un risultato esteticamente soddisfacente.

La via delle annotazioni ad un'immagine generata con strumenti esterni richiederà l'importazione dell'immagine bitmap, ad esempio nell'ambiente `pspicture` di PSTricks. Questo metodo è di pratico impiego e può avvalersi dell'uso di una griglia di riferimento temporanea opportunamente sovrapposta all'immagine originale al fine di posizionare le annotazioni nei punti appropriati. Tale metodo è da preferirsi certamente se il numero delle annotazioni è contenuto e se esse si limitano a semplici simboli o paragrafi. Esso diventa laborioso se l'illustrazione che si intende realizzare dovesse contenere molti simboli e l'indicazione di angoli, assi e rette di riferimento o di punti particolari in relazione agli oggetti presenti nella scena (ad esempio il baricentro di un velivolo oppure gli assi di cerniera delle superfici aerodinamiche di governo). In tal caso l'esperienza insegna che la realizzazione di annotazioni grafiche ben proporzionate e visivamente corrette non è immediata e richiede quasi certamente più iterazioni di *editing*-stampa-ispezione.

Un discorso analogo vale per la manipolazione di viste esportate in formato vettoriale, con annotazioni inglobate e manipolabili *a posteriori* secondo un dato protocollo. Anche in questo caso sono inevitabili alcune iterazioni prima di arrivare ad un risultato finale soddisfacente: almeno le dimensioni e le posizioni effettive delle annotazioni vanno controllate a valle della produzione finale del documento.

Si tenga presente inoltre che spesso si dà per scontato l'uso di strumenti di modellazione tridimensionale. Tuttavia, se si vogliono valutare correttamente i vantaggi di un dato metodo di lavoro per la produzione di illustrazioni, vanno certamente considerati i costi ed i tempi di apprendimento del software CAD che si intende usare. I costi nel caso di CAD commerciali sono alti. I software di modellazione 3D non commerciali più usati sono Blender e K-3D[3]: sebbene ancora nella fase *pre-release*, essi sono dotati di caratteristiche funzionali paragonabili a quelle di costose applicazioni concorrenti. Tutti i CAD, commerciali e non, hanno comunque delle interfacce utente ricche di funzionalità ed a volte estremamente complesse. Quelle di Blender e K-3D sono poi note per essere non del tutto convenzionali ed orientate ad utenti programmatori o esperti. Dunque, in ogni caso, i tempi di apprendimento di strumenti di modellazione e manipolazione di scene tridimensionali sono senz'altro non brevi.

6.2 Aspetti del metodo di lavoro proposto

L'uso di Sketch al contrario può essere appreso in tempi sorprendentemente brevi. Un utente mediamente esperto, in possesso di nozioni di Geometria Analitica, grazie soprattutto all'intuitività della sintassi ed agli ottimi esempi guidati forniti nel manuale d'uso, arriva ad apprendere il linguaggio di Sketch ed a realizzare una prima illustrazione di media complessità nel giro di 2 ore di lavoro. Uno studente universitario, già in grado di padroneggiare l'uso di LaTeX e dei programmi correlati, dovrebbe poter raggiungere lo stesso risultato in mezza giornata di lavoro.

Le figure 5, 6 e 7, concepite appositamente per questo articolo, hanno richiesto all'autore un'ora di lavoro complessivo. Le due illustrazioni di Figura 8, grazie alla piccola libreria messa a disposizione da De Marco (2007), hanno richiesto 10 minuti di lavoro. A giudizio dell'autore il tempo massimo impiegato per la preparazione di un'illustrazione o di un grafico di buona qualità da inserire in un lavoro scientifico è accettabile se va dai 30 ai 60 minuti. Se l'illustrazione deve essere inserita in un libro i tempi accettabili sono di un ordine di grandezza superiore. Pertanto i tempi di lavoro richiesti da Sketch sono molto bassi per illustrazioni semplici, e mediamente bassi per illustrazioni di complessità superiore. Tuttavia se l'illustrazione deve essere particolarmente ricca di oggetti e di annotazioni o di effetti di *shading*, è preferibile scegliere un metodo alternativo o affidarsi a professionisti.

Per quanto riguarda la sostenibilità del ciclo di lavoro con Sketch e LaTeX, sia gli utenti che fanno uso di IDE (*Integrated Development Environment*) dedicati, come WinEdt[4] o TexnicCenter[5], sia quelli che preferiscono usare *editor* generici e gestiscono le compilazioni con un Makefile non dovrebbero apportare sostanziali sconvolgimenti al loro paradigma di lavoro se non aggiungere un passo di compilazione in più (vedi Figura 4).

La fase fondamentale del lavoro con Sketch è costituita dall'*editing* di codice sorgente nel suo

3. `http://www.k-3d.org`
4. `http://www.winedt.com`
5. `http://www.texniccenter.org`

Tabella 1: Alcuni dati sui tempi di compilazione. Si considerano due tipologie di illustrazione, una semplice ed una alquanto complessa. Risultati ottenuti con un PC dotato di processore Pentium 4 a 3.2 GHz, 2 Gb di memoria RAM, Windows XP, MikTeX 2.6.

	Figura 7		
	`sk→tex`	`tex→dvi`	`dvi→ps`
tempi di calcolo (s)	0.015	2.23	0.31
dimensioni output (Mb)	0.007	0.025	0.050
	Figura 13		
	`sk→tex`	`tex→dvi`	`dvi→ps`
tempi di calcolo (s)	0.38	8.31	4.56
dimensioni output (Mb)	0.670	7.88	8.189

linguaggio di modellazione. In questa fase avere accesso ad un *editor* con evidenziazione della sintassi personalizzabile diventa di fondamentale importanza. L'autore ha creato e messo a disposizione il file `sketch.vim` nell'archivio `skthings.zip` (De Marco, 2007). Questo *add-on* permette agli utilizzatori di *Vim*[6] di scrivere sorgenti in linguaggio Sketch usufruendo del *syntax highlighting* sia per i costrutti specifici di Sketch che per quelli in linguaggio LaTeX inclusi nei frammenti `special`.

Per quanto riguarda i tempi di compilazione, questi dipendono dalla complessità della scena e dal numero di elementi che compongono ciascun oggetto. La compilazione con Sketch è tipicamente molto veloce, anche nei casi in cui si richiede di rimuovere le facce nascoste nella rappresentazione di oggetti molto complessi. Meno veloce è la compilazione da parte di LaTeX dell'output di Sketch. Si riportano in Tabella 1 i tempi macchina richiesti per produrre due illustrazioni di diversa complessità.

In alcuni casi, quando il carico computazionale è appesantito dalla presenza nella scena di uno o più oggetti di grandi dimensioni è necessario accrescere la dimensione del *buffer* di memoria principale di LaTeX rispetto al valore di default. Per esempio, l'illustrazione di Figura 13 è stata ottenuta passando alla riga di comando di LaTeX l'opzione `-mem-max=9000000`.

Nel manuale utente (Ressler, 2007b) si riporta il risultato di un test estremo a cui Sketch è stato sottoposto al fine di verificarne la robustezza e la velocità di esecuzione. Ne risulta che la compilazione del cosiddetto *Stanford Bunny*[7], un modello tipicamente utilizzato per la verifica di algoritmi di computer grafica, costituito da circa 70000 triangoli, ha richiesto circa 6 secondi[8]. Una larga parte del tempo di esecuzione è stata richiesta per la scrittura in output del codice LaTeX mentre la rimenente è stata consumata dall'algoritmo di rimozione degli oggetti in secondo piano.

6. `http://www.vim.org`
7. `http://www.cc.gatech.edu/~turk/bunny/bunny.html`
8. Non è specificato il tipo di computer utilizzato per la prova né il sistema operativo.

7 Conclusioni

In questo articolo si è presentato un possibile approccio alla produzione di illustrazioni di elevata qualità a scopo didattico, contenenti delle rappresentazioni prospettiche di scene tridimensionali. Uno degli scopi dell'articolo è quello di introdurre il programma Sketch, uno strumento relativamente recente ed ancora poco conosciuto nella comunità di utilizzatori LaTeX.

Mediante alcuni esempi guidati sono state presentate le caratteristiche principali del linguaggio di modellazione di Sketch nonché le sue potenzialità per la rappresentazione di scene complesse.

Infine sono stati discussi alcuni aspetti salienti del metodo di lavoro proposto: valutazione approssimativa dei tempi di apprendimento, dimensioni gestibili, tempi di calcolo.

Riferimenti bibliografici

Burkardt, J. (2007). IVCon 3D Graphics File Converter. URL `http://ivcon-tl.sourceforge.net`.

Calcara, M. (1988). *Elementi di Dinamica del Velivolo, Vol. I.* Edizioni CUEN, Napoli.

Casarosa, C. (2004). *Meccanica del Volo.* Edizioni Plus, Pisa.

De Marco, A. (2007). Modelli di velivolo in Sketch ed esempi di script. URL `http://www.dpa.unina.it/demarco/skthings.zip`.

Etkin, B. (1982). *Dynamics of Flight, Stability and Control.* John Wiley & Sons.

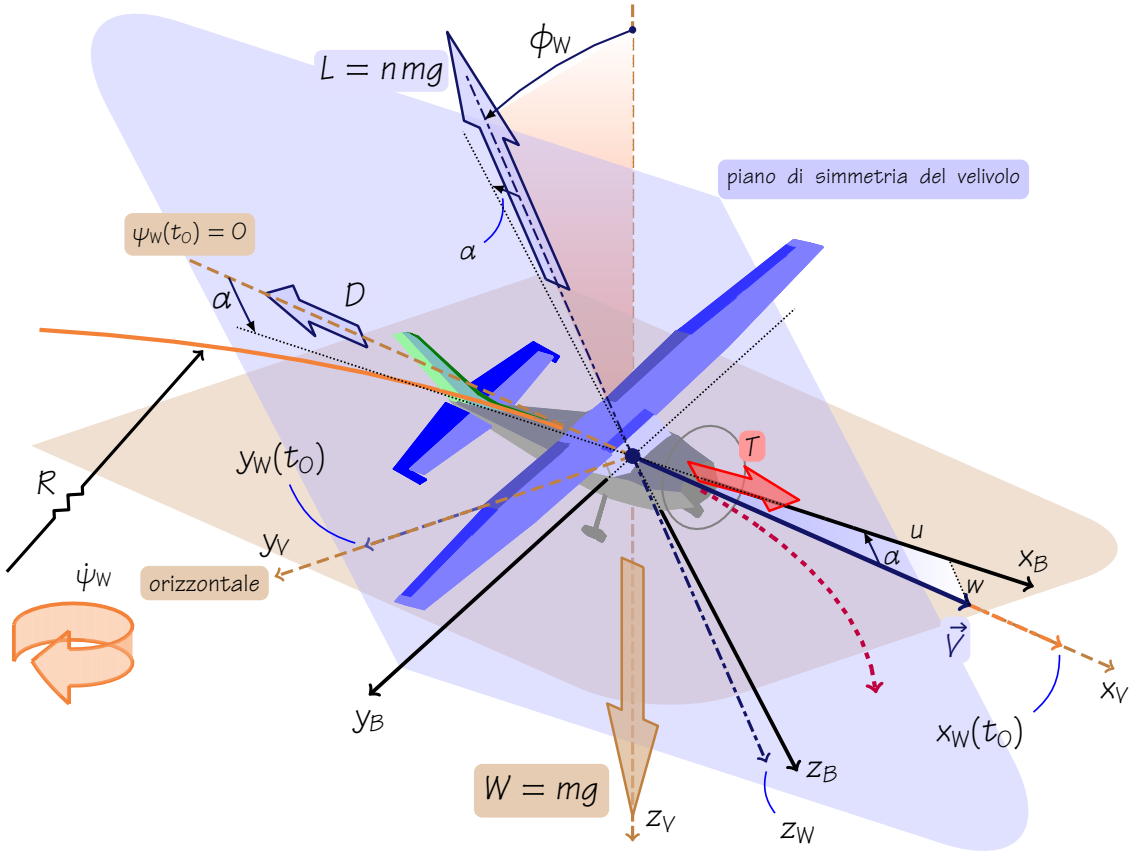


FIGURA 10: Definizione delle azioni esterne su di un velivolo in un'evoluzione di *virata corretta* a quota costante. Si evidenziano alcuni dei diversi sistemi di riferimento standard: il riferimento *verticale locale* (V, *local vertical*), quello degli *assi velivolo* (B, *body*) e quello degli *assi vento* (W, *wind*).

FAUSKE, K. M. (2007a). A PGF and TikZ examples gallery. URL `http://www.fauskes.net/nb/pgftikzexamples/`.

— (2007b). An introduction to Sketch 3D for PGF and TikZ users. URL `http://www.fauskes.net/nb/introduction-to-sketch/`.

— (2007c). Three dimensional graphics, illustrations and animations. URL `http://www.fauskes.net/nb/threedill/`.

MENGALI, G. (2001). *Elementi di Dinamica del Volo con Matlab*. Edizioni ETS, Pisa.

RESSLER, G. (2007a). Sketch website: Free Graphics Software for the TeX, LaTeX, and PSTricks Community. URL `http://www.frontiernet.net/~eugene.ressler`.

— (2007b). *Sketch. Simple 3D sketching*. URL `http://www.frontiernet.net/~eugene.ressler/manual.pdf`.

ROOSENDAAL, T. (2007). Blender website. URL `http://www.blender.org`.

SCHMIDT, L. V. (1998). *Introduction to Aircraft Flight Dynamics*. AIAA Education Series.

TANTAU, T. (2006a). PGF/Tikz website – Graphic systems for TeX. URL `http://sourceforge.net/projects/pgf`.

— (2006b). *The TikZ and PGF Packages*. URL `http://sourceforge.net/projects/pgf`.

TUG BOAT (2007). Pstricks website. URL `http://tug.org/PSTricks/`.

VAN ZANDT, T. (2003). *PSTricks. PostScript macros for Generic TeX*. URL `http://www.ctan.org/tex-archive/graphics/pstricks/base/doc/pstricks-doc.pdf`. Version 97.

▷ A. De Marco
Università degli Studi di Napoli "Federico II"
Dipartimento di Ing. Aerospaziale
`agostino.demarco@unina.it`

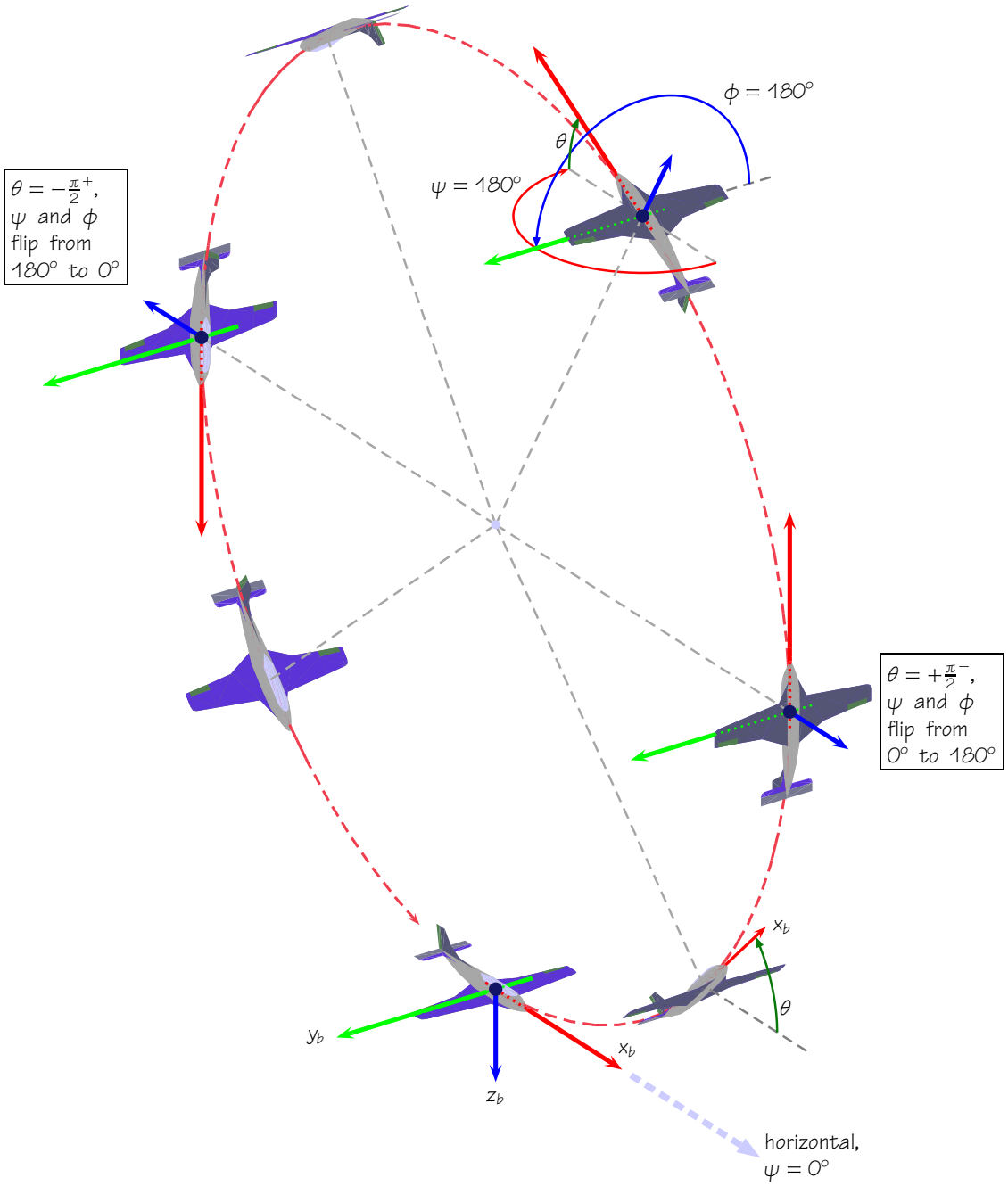
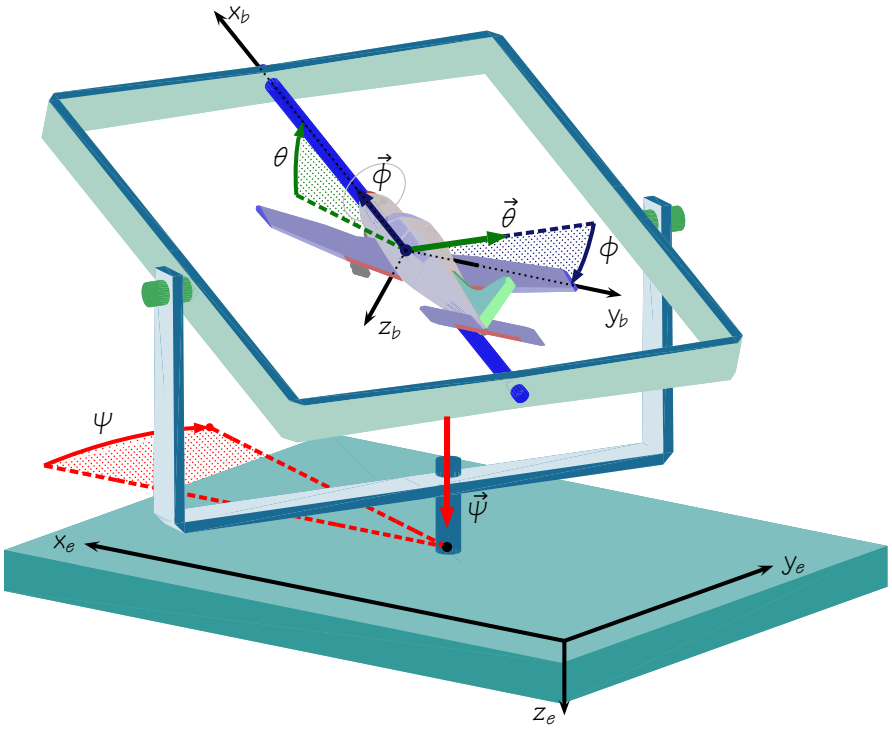
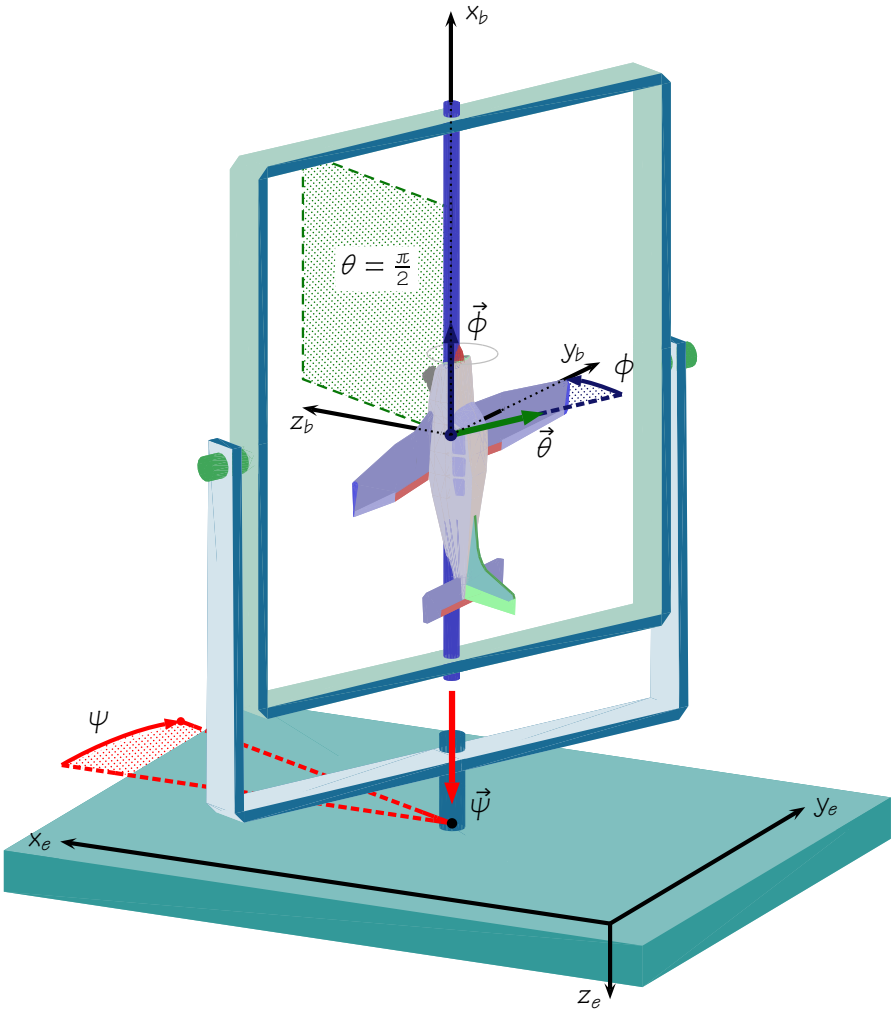


FIGURA 11: Un'illustrazione che mostra l'evoluzione di un velivolo detta *pull-up*. In un *pull-up* le storie temporali degli angoli di Eulero non sono continue.



(a) un orientamento generico



(b) l'assetto noto come *gimbal lock* in cui gli angoli di Eulero non sono definiti

FIGURA 12: Una classica illustrazione in cui l'orientamento del velivolo nello spazio, definito dagli angoli di Eulero, è visualizzato per mezzo di una sospensione cardanica.

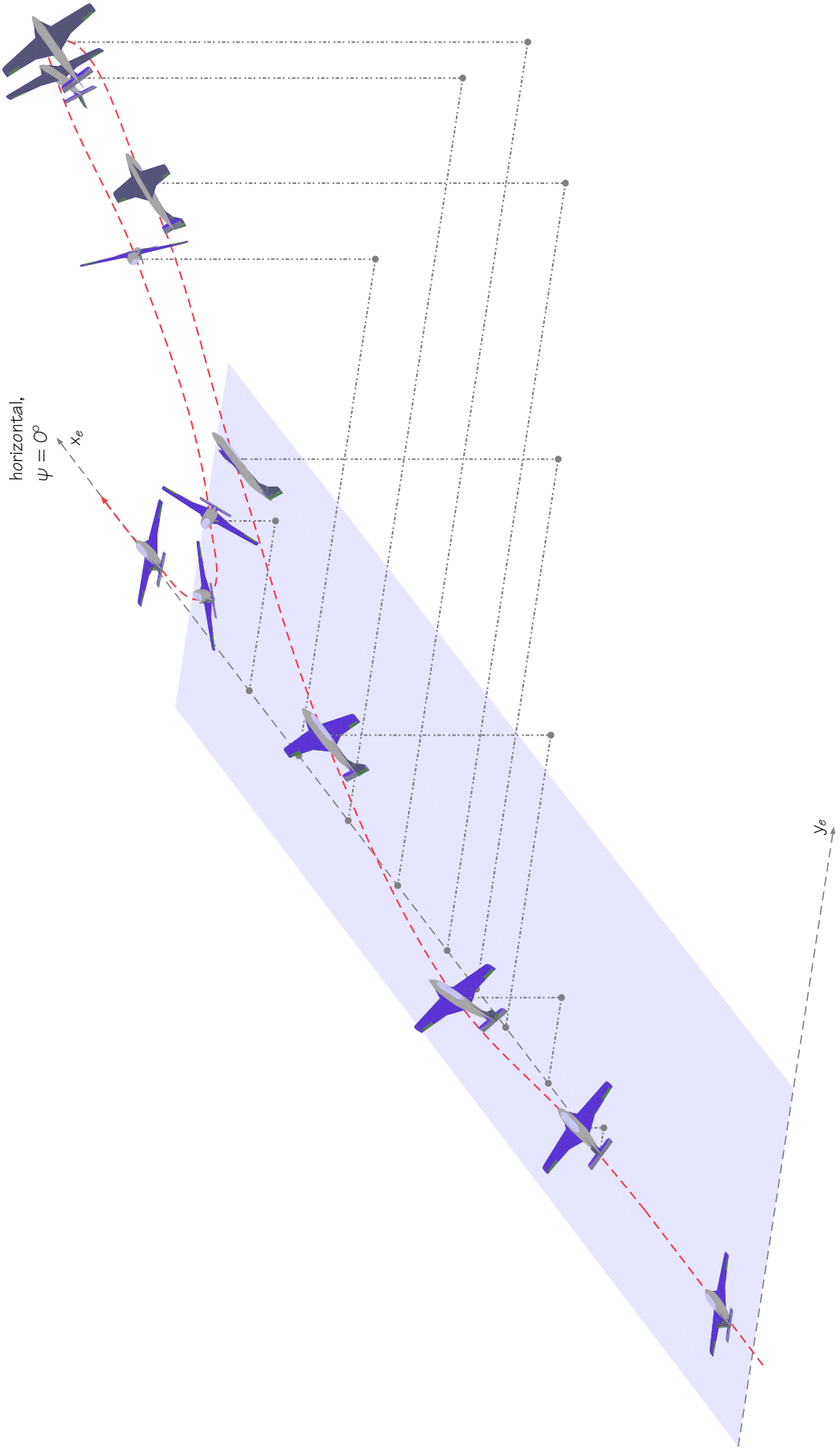


FIGURA 13: Esempio di evoluzione lungo una traiettoria non piana e con storie temporali non banali degli angoli di Eulero.

TEX Live's new infrastructure

Norbert Preining

Abstract

Since the release of TEX Live 2007 a new infrastructure for TEX Live distribution and management has been developed. This article presents the reasons for this switch, the ideas behind the new infrastructure, software developed, and ways to incorporate this new infrastructure. We will close with a look at what new features this new infrastructure could bring to the TEX (Live) world.

Sommario

Dopo l'uscita di TEX Live 2007 è stata sviluppata una nuova infrastruttura per la distribuzione e la gestione di TEX Live. L'articolo illustra le ragioni di questo cambio, le idee alla base della nuova infrastruttura, il software sviluppato e i modi per incorporarla. L'articolo chiude con uno sguardo alle nuove possibilità che la nuova infrastruttura introduce nel mondo di TEX (Live).

1 Introduction

TEX Live is an easy way to get up and running with TEX. It provides a comprehensive TEX system with binaries for most flavors of Unix, including GNU/Linux, and also Windows. It includes all the major TEX-related programs, macro packages, and fonts that are free software, including support for many languages around the world.

Since 1996 it tries to bring to all TEX-users as much material from CTAN as possible, packaged for 'consumption', i.e., for using it as a live system from DVD, or installing it on a variety of operating systems/architecture combinations.

These two requirements, incorporating as much as possible from CTAN, and providing binaries for a wide range of OS/arch combinations, has led to a huge number of supporting scripts, in a wide variety of programming languages (Perl, shell, XML, C, ...). These scripts were (and are) used to generate installation media, check consistency, update important files (e.g., for the installer), incorporating packages from CTAN, just to name a few. There have been many contributors; Sebastian Rahtz, Fabrice Popineau, and Karl Berry are the principal authors.

As always with overload volunteers working on big projects, not much was documented, programming was done by trial and error (see for example the packaging scripts of Debian, one of my own more horrible creations). This wouldn't have been reason enough to change things this deep in the intestines of TEX Live, but other illusions and dreams have driven us to rebuild from the ground up.

2 A world full of TPMs

Up to TEX Live 2007 everything in TEX Live was organized in `tpm`s, an acronym for 'TEX Package Manage[r/ment]'. One `tpm` described more or less one package from CTAN, containing the list of included files, title, description, license information, sometimes version numbers, additional information necessary for incorporation (activation of map files, hyphenation patterns, and formats).

Maintenance of these `tpm`s was mostly done by a Perl module `Tpm.pm` written by Fabrice Popineau, and the accompanying mystical program `tpm-factory.pl`, a script so full of possibilities that nobody besides Fabrice probably ever understood everything that could be done with it. Some of the jobs of this `tpm-factory.pl` were

- regeneration of the `tpm`s, this included some magic in finding the right files
- creation of a new `tpm` for a newly installed package from CTAN
- checking the coverage, i.e., checking that every file present in the repository is actually contained in a `tpm`.
- duplication and dependency checks

Alas, there have been some problems with all the `tpm`-business:

- they contained a mixture of generated (file lists) and static information (actions to be carried out);
- they were full of duplicate information: version, license, descriptions were taken now and then from the TEX Catalogue, but were typically horribly out of date or otherwise wrong;
- we had to generate so-called 'lists' files (line-oriented plain text file lists) from the `tpm`s for the installer because the XML syntax of the `tpm`s is not practical to parse from a simple shell script.

3 Aims of the new infrastructure

So around mid-2006 discussion of a new infrastructure started, but unfortunately, due to the usual

time constraints of all involved, without much concrete outcome except many emails. At least a preliminary catalogue of items to be improved sprang into existence:

Separation of static from generated content

As already mentioned, the `tpm`s contained a wild mixture of stuff from various sources (the file itself, the TEX Catalogue, the repository). The new infrastructure should have some kind of 'source' files where *only* the necessary information is stored, and all other content is added at generation time (of whatever). Naturally that could have been done on top of the `tpm`s, too, but starting from scratch seemed to be a better option.

Getting rid of lists files

Using the `tpm`s and some XSLT processing, the lists files were generated before release. These lists files (one for each package, about 2000 in all) must be read by the installer from the DVD, which caused a lot of headache and slow installations.

The sole reason for the existence of these lists files was the fact that the installer, written in shell, couldn't parse the XML `tpm`s, since XML parsing programs cannot be assumed to be present at all installations. The new infrastructure should be based on some format that is easily parseable using stuff normally already present, or at least necessary for TEX Live anyway.

Single package updates via the web

Some TEX distributions, notably MiKTeX, already support single package updates over the Internet. TEX Live, as big and nice as it is, still lacks this lovely feature. Of course it was put high on the priority list to allow for single package updates.

Better documentation

Last, but not least, we hoped that by rewriting the infrastructure and documenting it on the way we could provide a more stable foundation for more development and improvement, thus attracting more contributors. Furthermore, since teTEX development has been stopped, TEX Live is taking the place as the TEX system of choice in many GNU/Linux and other free distributions, and better documentation would only help providing those distributions with some aid for the migration.

4 New infrastructure – the basics

There are three type of files: `tlpsrc`, `tlpobj`, `tlpdb`. The format of these files are very similar to Debian's Package files: a simple list of

```
key    value
```

(without leading spaces). Empty lines at the beginning and end of a `tlpsrc` or `tlpobj` file are ignored. Also lines starting with `#` are ignored (to allow for comments). Some special cases apply for `tlpobj` files and the `tlpdb` file, see below.

4.1 `tlpsrc` file format

The possible keys and their respective interpretation for the `tlpsrc` files are:

name identifies the package, `value` must consist only of `[-_a-zA-Z0-9]`.

category identifies the category into which this package belongs. Possible categories are `Collection`, `Scheme`, `TLCore`, `Documentation`, `Package`. There are no particular checks as to whether a `tlpsrc` file called `collection-something` actually belongs to the category `Collection`. Most packages will fall into the `Package` category. These categories were inherited from the `tpm` world and may be modified at some point; for now, they suffice.

catalogue identifies the name under which this package can be found in the TEX Catalogue. If not present the name of the packages is taken as the Catalogue entry.

shortdesc gives a one line description of the package. Susequent entries will overwrite the former ones. In TEX Live only used for collections and schemes.

longdesc gives a long description of the package. Susequent entries are added up to a long text. In TEX Live only used for collections and schemes.

depend gives the list of dependencies of the package in the form

```
Category/Name
```

for `value`. All the depend lines contribute to the dependencies of the package.

execute gives a free form entry of install time jobs to be executed. Currently the following values are understood by the installers:

- `execute addMap font.map` enables the font map file `font.map` in the `updmap.cfg` file.
- `execute addMixedMap font.map` enables the font map file `font.map` for Mixed mode in the `updmap.cfg` file. By the way, the purpose of MixedMap is to help users with printers that render the Type 1 versions of the fonts worse than the (mode-tuned) bitmap-based Type 3 fonts. The entries from MixedMap are not added to psfonts_pk.map; that's the only difference. It's used for fonts which have both Metafont and (typically) autotraced Type 1 instantations.

- `execute BuildLanguageDat NN` activates all the hyphenation patterns found in `Master/texmf/tex/generic/config/language.NN.dat` in the generated `language.dat` file.
- `execute BuildFormat FMTCFG` activates all the formats present in `Master/texmf/fmtutil/format.FMTCFG.cnf` in the generated `fmtutil.cnf` file.

(src|run|doc|bin)pattern pattern adds a pattern (see below) to the respective list of patterns.

4.1.1 Patterns

To automatically generate the file lists in `tlpobj`s a specific pattern language was developed. Patterns can include or exclude files or whole sub-directories into the file list of a `tlpobj`. Patterns are of the form

```
[PREFIX]TYPE PAT
```

where `PREFIX` can be `+`, `!+`, or `!`, and `TYPE` one of the letters `t`, `f`, `d`, `r`. An initial `+` for the pattern indicates that the automatically generated pattern should not be cleared (see below), while the `!` indicates that the matching files should be excluded.

The meaning of the various pattern types is

f path includes all files which match `path` where *only* the last component of `path` can contain the usual glob characters * and ? (but no others!).

d path includes all the files in and below the directory specified as `path`.

t word1 ... wordN wordL includes all the files in and below all directories of the form

```
word1/word2/.../wordN/.../any/dirs/../wordL/
```

i.e., all words but the last form the prefix of the path, then there can be an arbitrary nesting of directories, followed by `wordL` as another directory.

r regexp includes all files matching the Perl regexp `/^regexp$/`

4.1.2 Autogenerated patterns

In the case that one of the pattern sections is empty or *all* the provided patterns have the prefix `+` (e.g., `+f ...`), then the following patterns are *automatically* added at expansion time (but never written to the textual representation):

- for runpatterns of category `Package`

```
t texmf-dist topdir name
```

where `topdir` is one of: bibtex, context, dvips, fonts, makeindex, metafont, metapost, mft, omega, scripts, tex, vtex. `name` refers to the name of the package as given by the `name` directive.

For other categories *no* patterns are automatically added to the list of runpatterns.

- for docpattern of category `Package`

```
t texmf-dist doc name
```

and for docpattern of category `Documentation`

```
t texmf-doc doc name
```

- for srcpattern of category `Package`

```
t texmf-dist source name
```

and for srcpattern of category `Documentation`

```
t texmf-doc source name
```

`binpatterns` are never automatically added.

In addition some magic tricks have been added to make writing of `binpatterns` easier:

arch expansion In case the string `${ARCH}` occurs in one `binpattern` it is automatically expanded to the respective architecture.

bat/exe/dll for win32 For `binpatterns` of the form `f bin/win32/foobar` (i.e., also for a binpattern of the form `f bin/${ARCH}/foobar`) files `foobar.bat`, `foobar.dll`, and `foobar.exe` are also matched.

The above two properties allows to capture the binaries for all architectures in one binpattern:

```
binpattern f bin/${ARCH}/dvips
```

will include `bin/win32/dvips.exe` in the runfiles when arch=win32.

Note that the bat/exe expansion *only* works for patterns of the `f`-type.

4.2 `tlpobj` file format

These files are structurally similar to `tlpsrc` files; the only difference is that the `*pattern` keys are not allowed, their place being taken by `*files` keys having a peculiar syntax (see below). Furthermore, an additional key `revision` and all keys matching `catalogue-*` are allowed. The `catalogue-*` keys specify information simply copied from the TEX Catalogue. We'll discuss the others below.

4.2.1 The `revision` entry

The `revision` key is *not* related to the package's version as specified by the author. Those versions are far too random, hard to extract, and sometimes not even present, to be a reliable basis for a packaging system. Instead, we look at all the package's files in the TEX Live source repository, which is currently Subversion, and use the maximum version number found there. Since Subversion uses simple integers for version numbers, this is simple to use in programs, and just as important, this whole process of finding the version number can be reliably automated.

4.2.2 The `*files` entries

While the `tlpsrc` files specify the files to be included using the patterns, the `tlpobj` files contain the already expanded file list. All the 'files' keys have in common that they are followed by a list of files (and possibly additional tags) indented by exactly one (1) space. They differ only in the first line itself (described below).

srcfiles, runfiles each of these lines is (or better should be) tagged with `size=NNNN` where `NNNN` is the sum of sizes of the single files (currently in bytes), e.g.,

```
srcfiles size=NNNNNN
```

docfiles The docfiles line itself is similar to the `srcfiles` and `runfiles` lines above:

```
docfiles size=NNNNNN
```

But the lines listing the files are allowed to have additional tags `details` and `language`:

```
file detail="foo bar" language="en"
```

For an example see listing 1. The reason to add these tags is the hope that someone will write a nice replacement for `texdoctk`. Note that these tags' source is the TEX Catalogue, and are in no way obligatory.

binfiles Since `binfiles` are different for the different architectures, one `tlpobj` file can contain `binfiles` lines for different architectures. The architecture is specified on the `binfiles` lines using the `arch=XXX` tag. Thus, `binfiles` lines look like

```
binfiles arch=XXXX size=NNNNN
```

A more complete example taken from `bin-dvipsk.tlpobj` can be seen in listing 2.

Listing 1: Excerpt from achemso.tlpobj

```
...
docfiles␣size=135842
␣texmf-dist/doc/latex/achemso/⤸
    ⤷README␣details="Package␣⤸
    ⤷Readme"␣language="de"
␣texmf-dist/doc/latex/achemso/⤸
    ⤷achemso.pdf␣details="⤸
    ⤷Package␣documentation"␣⤸
    ⤷language="de"
␣...
```

Listing 2: Excerpt from bin-dvipsk.tlpobj

```
name␣bin-dvipsk
category␣TLCore
revision␣4427
docfiles␣size=959434
␣texmf/doc/dvips/dvips.html
␣...
runfiles␣size=1702468
␣texmf/dvips/base/color.pro
␣...
␣texmf/scripts/pkfix/pkfix.pl
binfiles␣arch=i386-solaris␣size⤸
    ⤷=329700
␣bin/i386-solaris/afm2tfm
␣bin/i386-solaris/dvips
␣bin/i386-solaris/pkfix
binfiles␣arch=win32␣size=161280
␣bin/win32/afm2tfm.exe
␣bin/win32/dvips.exe
␣bin/win32/pkfix.exe
...
```

4.3 `tlpdb` file format

A `tlpdb` file will describe the status of an installation, that is it will contain all the installed packages' `tlpobj` files. Thus, the format of a `tlpdb` file is quite simple. It is the concatenation of `tlpobj`s with (at least) one empty line between different `tlpobj`s.

5 Programming APIs

Wrapping these file formats into a programming API represents the actual work of realizing the new infrastructure. On the way we had to create not only modules for accessing the content of the above files in some object-oriented way, we also created some additional modules for interfacing to the Subversion repository and the TEX Catalogue.

Currently, the following modules are available in the TEX Live subversion repository:

TeXLive::TLTREE an object that exhibits the properties of the subversion repository in some structured form. In principle it is the scooping up of `svn status -v` output with post-processing for easier and faster searches.

TeXLive::TeXCatalogue a very simple interface to the TEX Catalogue, barely providing minimal information. Will be used for enriching the final database.

TeXLive::TLPSRC provides access to `tlpsrc` files, basic functionality like reading in, writing out, and member access functions. In addition, it allows to generate, or expand, a `TLPSRC` object to a `TLPOBJ` object using an instance of `TLTREE`. That is, it takes the patterns specified in the `tlpsrc`, and tries to find all files matching these patterns. After this it recomputes the size and returns a `TLPOBJ` object.

TeXLive::TLPOBJ provides access to `tlpobj` files, and entries stored in a `TLPDB` object. As with the `tlpsrc`, basic functionalities are exported, and also a function to create a zip file for the `TLPOBJ` (which later should be used for web updates or building the 'inst' CD), and together with an instance of a `TeXCatalogue` object some information can be transferred from the TEX Catalogue to the `TLPOBJ` object.

TeXLive::TLPDB provides access to the TEX Live database. These files will serve the central purpose of describing the current status of various media, like an installation on a user computer, or the distribution. Comparing these status descriptions, update programs can deduce what packages should be updated.

TeXLive::TLPUtils exports some handy functions used in all the other modules.

The full API documentation is provided in `pod` format within the perl modules. A Perl API document is available in the TEX Live subversion repository, too.

In addition to the Perl API a start at a shell (sh, bash, etc.) API and implementation has been created, but it presently lacks several key features, and is not up to date. Jim Hefferon has contributed a start at a Python API. However, we assume that on all computers Perl will be available or will be installed during installation.

6 Integration into current TEX Live development

Around these modules several day-in-day-out tasks of the TEX Live developers have been rewritten using the new infrastructure, the most important one being the script `ctan2tl`. This is the script responsible for transferring a package from CTAN into the right places in the TEX Live repository, and preparing it for users.

Other things already converted are automatic update of the `texlive.tlpdb`, the file containing the `TLPDB`.

On the other hand not everything has been done, the biggest piece for sure is the rewrite of the installer. The old installer, relying on the lists files we wanted to get rid of, is still the only one we have. Furtunately we can generate lists files from the `texlive.tlpdb`, but in the not-so-long run, meaning hopefully for TEX Live 2008, we want to have a new installer.

Other things still missing are coverage and duplication checks, but those are quite easy to do using the Unix command world (`grep`, `sort`, and `uniq` being the magic incantations).

7 Closing

Although we are quite confident that the new infrastructure will work well, only the reality of the next release will prove us right or wrong. Many of the key features of the new infrastructure have been designed with the `tpm`-system in mind, and some 'features' carried over will probably proven superfluous. Nothing with the current infrastructure is written in stone, and we are open for proposals and improvements, keeping in mind that releasing TEX Live should not be made more difficult.

That said, we invite contributors and everyone interested to contact us at `tex-live@tug.org`, or take a look at the following resources:

- `http://www.tug.org/texlive` – the main entry point, with links to developers' resources, documentation;
- `http://www.tug.org/svn/texlive/trunk/` – web view onto the subversion repository;
- `svn://tug.org/texlive/trunk` – svn repository, anonymous access, take care when you checking out the repository, it needs several Gigabyte of disk space;
- `http://www.tug.org/texlive/pkgupdate.html` – an explanation how updates from CTAN to TEX Live are done.

Finally, I want to thank all the contributors to TEX Live, there are too many to mention. In the course of developing this infrastructure the discussions with Karl Berry, Paweł Jackowski, Reinhard Kotucha, Siep Kroonenberg, and Jerzy B. Ludwichowski were very helpful.

▷ Norbert Preining
Vienna University of Technology
Wiedner Hauptstr. 10
1040 Wien, Austria
`preining@logic.at`

Typesetting tables with LaTeX

Klaus Höppner

Sommario

From a LaTeXoligist's point of view, LaTeX is a perfect tool to typeset nearly everything in a beautiful manner. Without any doubt, LaTeX can typeset tables, but it is easy to produce bad tables with ugly lines and text touching the lines. This talk is intended to introduce how to typeset tables with LaTeX on a beginners' level, mentioning some typographic aspects, showing some packages that help the author in formatting tables and concluding with how to typeset tables with page breaks.

Sommario

Dal punto di vista di un LaTeXologo, LaTeX è uno strumento perfetto per comporre praticamente qualsiasi cosa con risultati esteticamente gradevoli. Senza dubbio, LaTeX può comporre tabelle, ma è molto facile produrre brutte tabelle con linee orribili e il testo che tocca le linee. Questo intervento è mirato a illustrare come produrre tabelle con LaTeX a livello di principiante, citando alcune particolarità tipografiche, mostrando alcune estensioni che possono aiutare l'autore nel dar forma alle tabelle e concludendo con la maniera di comporre tabelle che si estendono per più pagine.

1 Basic tables

LaTeX already has built-in support to typeset tables. For beginners it may be a bit confusing, since LaTeX provides two environments: tabular and table. To typeset material in rows and columns, tabular is needed, while the table environment is a container for floating material similar to figure, into which a tabular environment may be included.

So, let's have a look how to typeset a simple table:

```
\begin{tabular}{lcr}
a   & b   & c\\
aa  & ab  & ac\\
aaa & aab & aac
\end{tabular}
```

will result in

a	b	c
aa	ab	ac
aaa	aab	aac

The rows of the table are divided by LaTeX's usual `\\` command (in some cases, it may be needed to use `\tabularnewline` instead, as we will see later in this article). Columns are separated by `&`, the ampersand character.

The required argument of tabular defines the basic layout of the table, especially the alignment of the columns:

l left aligned column

c centered column

r right aligned column

p{⟨*width*⟩**}** paragraph-like column of a predefined width (with the baseline of the paragraph's first line aligned relative to the other cells in the table row)

The normal space between columns, which is also added before the first and after the last column, may be overridden by `@{`⟨*sep*⟩`}`, where ⟨*sep*⟩ is any LaTeX code, inserted as the separator. For illustration, let's typeset some flight data:

flight no.	route
LH 402	Frankfurt–Newark
KL 3171	Amsterdam–Cork
US 1152	San Diego–Philadelphia

Here, the `@` command is used twice: The space that normally would have been inserted left of the first column is replaced by nothing, thus the table is left aligned with the surrounding text (compare it with the first tabular example in this article, you will see the difference). Additionally, the inter-column space between the points of departure and destination is replaced by a dash. So the code used to produce this table looks as follows (silently introducing `\multicolumn` to combine cells):

```
\begin{tabular}{@{}lr@{--}l}
flight no. & \multicolumn{2}{c}{route}\\
LH\,402 & Frankfurt & Newark\\
KL\,3171 & Amsterdam & Cork\\
US\,1152 & San Diego & Philadelphia
\end{tabular}
```

2 Extra packages for typesetting tables

Beyond LaTeX's built-in ability to typeset tables, several extra packages exist. Some of them add new effects in typography and layout, others simplify the task of writing the document's source code. The packages that I will introduce in this article (and more that I won't) are covered in detail in the *LaTeX Companion* by Mittelback *et al.* (2004).

Here are some important packages for authors who want to typeset tables:

array adds paragraph-like columns `m{`⟨*width*⟩`}` and `b{`⟨*width*⟩`}` similar to the `p`-column, but vertically aligned to the center or bottom. Additionally, the package allows defining command sequences to be executed before or after the contents of a column.

tabularx typesets a table with fixed widths, introducing the column type `X` that works like a `p`-column with automatically calculated width.

booktabs provides fancy commands for horizontal lines with appropriate spacing above and below.

We'll now briefly discuss each of these in turn.

2.1 Using array

From my point of view, the most important feature of array written by Mittelbach e Carlisle is the new possibility of defining commands that are added automatically before or after a column's cell. It saves a lot of typing, making the source code more concise and more flexible. array is one of the required LaTeX packages, so it must be part of any LaTeX installation.

For a simple example, have a look at the following table:

Command	Symbol
\alpha	α
\beta	β
\gamma	γ

The left column displays LaTeX commands, beginning with a backslash and using a typewriter font, while the right columns displays the corresponding math symbol. Using the array package, the source code is pretty straightforward:

```
\begin{tabular}%
    {>{\ttfamily\textbackslash}c>{$}c<{$}}
\multicolumn{1}{c}{Command} &
    \multicolumn{1}{c}{Symbol}\\
\hline
alpha & \alpha\\
beta & \beta\\
gamma & \gamma
\end{tabular}
```

As shown in this code, we can now define a command sequence inside the `>{...}` preceding the column type definition. These commands are executed before each cell of that column. Similarly, using `<{...}` after the column type defines which commands to be executed after typesetting the column's cells.

In the example above, the first row is different from the others, since it contains the column titles that obviously should not be typeset in typewriter font or math mode. This is handled by 'abusing' the `\multicolumn` command, which prevents the `>` and `<` command hooks from being applied for these cells.

Another use of these command hooks is typesetting paragraphs in narrow columns. LaTeX typesets these paragraphs left and right justified by default, but in narrow columns it is often more appropriate to typeset them using `\raggedright`. So we might think of trying the following code:

```
\begin{tabular}{l>{\raggedright}p{3in}}
```

Unfortunately this fails when ending the table rows with the `\\` command, with rather weird error messages about misplaced aligns. The problem is that `\raggedright` redefines `\\`, so it can't be recognized as the end of table rows. There are three solutions for this problem:

1. Use `\tabularnewline` instead of `\\`. In fact, it does no harm to always use this, even when you don't have problems with `\raggedright`.

2. Restore the original definition of `\\` by using the command `\arraybackslash`, as follows:

   ```
   \begin{tabular}%
       {l>{\raggedright\arraybackslash}p{3in}}
   ```

3. Use Martin Schröder's ragged2e package. It redefines the command `\raggedright` to prevent it from redefining `\\`, so the problem disappears without any further change to the original code. Additionally, ragged2e provides the new command `\RaggedRight` that typesets the paragraph left aligned, but doesn't disable hyphenation.

2.2 Using tabularx

Besides the normal tabular environment, a rarely used environment tabular* exists. In addition to the column definition, it takes a table width as argument. The resulting table is typeset to this width, but surprisingly by expanding the space between columns.

A more convenient implementation is done by the tabularx (Carlisle, d) package (another required LaTeX package present in every LaTeX installation). This introduces a column type `X` for paragraph-like columns whose width is automatically calculated in order to achieve a table of a desired total width. For example, let's look at the following:

```
\begin{tabularx}{\linewidth}{lX}
Label & Text\\
\hline
One & This is some text without meaning,
  just using up some space. It is not
  intended for reading.\\
...
\end{tabularx}
```

This produces a table across the full line width, where the right column just uses the space remaining after typesetting the left column:

Label	Text
One	This is some text without meaning, just using up some space. It is not intended for reading.
Two	This is another text without meaning, just using up some space. It's not intended for reading either.
Three	This is yet another text without meaning. Guess what? It's not intended for reading. It is just there.
Four	How often did I mention that you should not read this text?

It is possible to use more than one `X`-column. By default, all of them are typeset to the same width, but it is possible to manually adjust how the available space is divided. Here's our next example:

Label	Text	More text
One	This is some text without meaning.	This is another text without meaning, just using up some space. It is not meant for reading either.

This table was produced with the following code:

```
\begin{tabularx}{\linewidth}%
 {l>{\setlength\hsize{0.6\hsize}\raggedright}X%
   >{\setlength\hsize{1.4\hsize}\raggedright}X}
Label & Text & More text\tabularnewline
\hline
...
\end{tabularx}
```

When balancing the column widths manually, it is important that the `\hsize` fractions add up to the number of `X`-columns, as in the example above, where $0.6 + 1.4 = 2$. To achieve *automatic* balancing of columns, take a look at the tabulary package.

Be aware that the way tabularx parses the contents of a table limits the possibility of defining new environments based on the tabularx. If you consider doing this, first look at the documentation.

3 Using lines in tables

LaTeX provides the possibility of using lines in tables: vertical lines are added by placing a `|` at the appropriate position in the definition of the column layout, and horizontal lines are added by using the command `\hline`.

While using lines in tables can help the reader in understanding the contents of a table, it is quite easy to produce really ugly tables like the following:

Label	Text	More text
One	This is some text without meaning.	This is another text without meaning, just using up some space.

Though nobody would typeset this particular table in real life, it illustrates a general and common problem: the column titles and the word "another" in the rightmost column touch the horizontal lines above them.

As a first step to improve the spacing between the table rows and the horizontal lines in such cases, set `\extrarowheight` to a non-zero length, e. g. to 4 pt. If this isn't enough, additional adjustment may be done by adding invisible rules. Here is revised source code for the above example illustrating both these points:

```
\setlength{\extrarowheight}{4pt}
\begin{tabularx}{\linewidth}%
    {|l|>{\setlength\hsize{0.67\hsize}}X%
       |>{\setlength\hsize{1.33\hsize}}X|}
\hline
\Large Label & \Large Text
    & \Large More text\tabularnewline
\hline
\hline
One & This is some text without meaning.
    & \rule{0pt}{18pt}%
      This is {\huge another} text without meaning,
      just using up some space.\\
\hline
\end{tabularx}
```

we get a somewhat better result:

Label	Text	More text
One	This is some text without meaning.	This is another text without meaning, just using up some space.

Please notice that the `\rule` used as an additional spacer was typeset with a horizontal width of 0.4 pt instead of 0 pt (as shown in the code) in order to make its effect and location visible.

Even after this, the layout of the table still looks quite poor, e. g. the broken vertical lines between the double horizonal line. This might be solved with the package hhline (Carlisle, a), but for typesetting tables with pretty lines, have a look at the booktabs package by Fear e Els. It starts by giving users a basic piece of advice, namely to avoid vertical lines, and introduces commands to typeset horizontal lines with appropriate thickness and spacing. Using booktabs, the source code for our weird example now looks as follows:

```
\begin{tabularx}{\linewidth}%
    {l>{\setlength\hsize{0.67\hsize}}X%
      >{\setlength\hsize{1.33\hsize}}X}
\toprule
\Large Label & \Large Text
```

```
    & \Large More text\tabularnewline
\midrule
One & This is some text without meaning.
    & This is {\huge another} text without meaning,
      just using up some space.\\
\bottomrule
\end{tabularx}
```

Using this, the result becomes:

Label	Text	More text
One	This is some text without meaning.	This is another text without meaning, just using up some space.

At last, we've improved the layout of the table quite a bit. The content with arbitrary changes of font size still looks weird, but that's something for which the author and not LaTeX must be blamed.

For a more realistic example of using rules, here I present an example from the booktabs manual:

	Item	
Animal	Description	Price ($)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

4 Typesetting tables across multiple pages

The usual tabular(x) environment is restricted to single-page tables, i. e. no page breaks are allowed within tables.

However, two extension packages provide support for typesetting tables across multiple pages, namely longtable (Carlisle, b) and supertabular (Braams e Jurriens). The main difference between them is that longtable keeps the column widths the same throughout the entire table, while supertabular recalculates the column widths on each page. According to the documentation, longtable doesn't work in two- or multi-column mode, and I didn't try supertabular on this case. The syntax of the two packages is different, so one has to decide which one to use.

Let's have a look at the general structure of a longtable:

```
\begin{longtable}{ll}
Label (cont.) & Text (cont.)\\
\endhead
\multicolumn{2}{l}{This is the first head}\\
Label & Text\\
\endfirsthead
\multicolumn{2}{l}{to be cont'd on next page}
\endfoot
\multicolumn{2}{l}{this is the end (finally!)}
\endlastfoot
One & Some content\\
Two & Another content\\
Three & Yet another content\\
[...]
\end{longtable}
```

As shown in this source code, the longtable environment may contain definitions for headers and footers before the normal content of the table:

\endfirsthead defines what to typeset at the very first head of the table,

\endhead defines a table head that is used on continuation pages,

\endfoot defines what to typeset on the foot of the table if a continuation page follows, and

\endlastfoot defines the foot at the very end of the table.

I personally prefer longtable simply because there exists yet another package ltxtable (Carlisle, c), which combines longtable and tabularx. It provides the command `\LTXtable{`⟨*width*⟩`}{`⟨*file*⟩`}`, which reads a given file containing a longtable environment using `X`-columns and typesets it to the requested width.

Riferimenti bibliografici

Braams, J. e Jurriens, J. «The supertabular package». CTAN: `macros/latex/contrib/supertabular`.

Carlisle, D. (a). «The hhline package». CTAN: `macros/latex/required/tools`.

— (b). «The longtable package». CTAN: `macros/latex/required/tools`.

— (c). «The ltxtable package». CTAN: `macros/latex/contrib/carlisle`.

— (d). «The tabularx package». CTAN: `macros/latex/required/tools`.

Fear, S. e Els, D. «The booktabs package». CTAN: `macros/latex/contrib/booktabs`.

Mittelbach, F. e Carlisle, D. «The array package». CTAN: `macros/latex/required/tools`.

Mittelback, F., Goossens, M. *et al.* (2004). *The LaTeX companion.* Addison Wesley, 2ª edizione.

Schröder, M. «The ragged2e package». CTAN: `macros/latex/contrib/ms`.

▷ Klaus Höppner
Haardtring 230 a
64295 Darmstadt
Germany
`klaus.hoeppner (at) gmx (dot) de`

Inserire equazioni LaTeX in grafici di *Mathematica*

G. Gorni, S. Matiz

Sommario

Creare dei grafici da inserire nei propri articoli LaTeX o TeX rappresenta certamente un problema soprattutto quando, per mantenere l'omogeneità dei simboli, si vogliono usare le font LaTeX anche all'interno del grafico. In questo articolo presentiamo una soluzione per coloro che usano e conoscono un po' la sintassi del software *Mathematica* e che conoscono LaTeX.

Con un unico comando, all'interno dell'interfaccia *Mathematica*

`TeXClipping[`*sintassi LaTeX*, *opzioni*`]`

si ottiene un oggetto grafico `Graphics` per *Mathematica*: internamente è un insieme di poligoni e da un punto di vista visivo, non ha nulla da invidiare al suo collega font e si integra perfettamente nei grafici del suo ambiente.

Abstract

In this article we introduce a solution for creating graphics with the LaTeX fonts. This solution is meant for users that create pictures in *Mathematica* to be included in LaTeX documents. With a single command, within the *Mathematica* front end

`TeXClipping[`*LaTeX syntax*, *options*`]`

we get a graphical object `Graphics` for *Mathematica*: a simple set of polygons that gives the same impression of its colleague the font and that integrates perfectly in the *Mathematica* ambient.

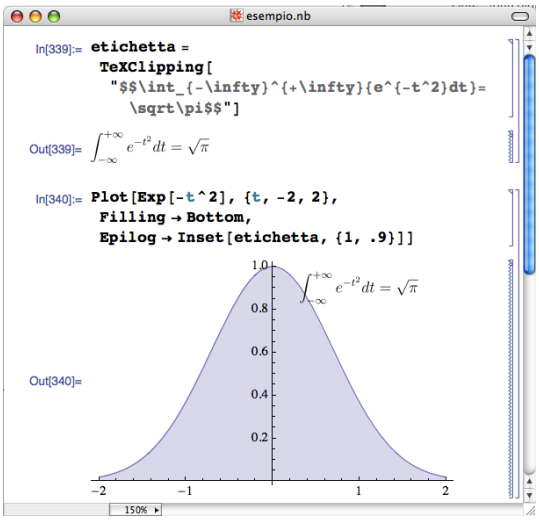


Figura 1: uno screenshot di *Mathematica* al lavoro

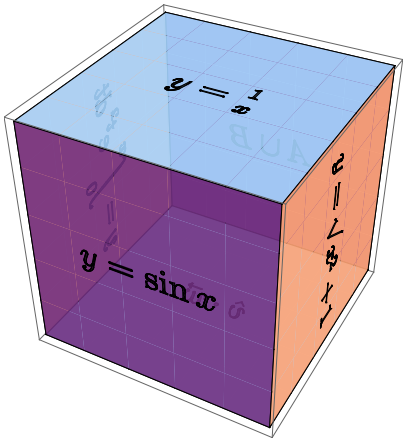


Figura 2: dado matematico

1 Introduzione

Il nostro proposito è stato quello di creare un sistema per gli utilizzatori di *Mathematica* che consentisse la creazione di grafici da inserire nei propri articoli LaTeX o TeX, con le font di TeX. Il sistema è di semplice utilizzo, non è necessario imparare nuovi linguaggi di programmazione e si integra perfettamente nell'ambiente di lavoro dando per esempio la possibilità in *Mathematica* versione 6 di copiare, incollare e spostare le formule LaTeX con il mouse all'interno di un grafico.

Sul web si possono trovare parecchie soluzioni per creare dei grafici con le font LaTeX da inserire nei propri articoli. Qui di seguito citiamo alcune tra le più diffuse

- PSTricks è un linguaggio di programmazione con sintassi affine a LaTeX che permette di creare grafici e disegni molto avanzati e usa naturalmente le font LaTeX.
- WARMreader, di Ross Moore, permette di inserire delle etichette e delle annotazioni all'interno di figure e grafici importati in documenti TeX e LaTeX.
- Gnuplot è un motore matematico che permette di trasformare il grafico prodotto in codice LaTeX pronto per essere compilato.
- METAPOST, Ti*k*z e XY-pic sono altri linguaggi di programmazione per il disegno simili a LaTeX ed in cui si possono usare le font LaTeX di default.

Il software commerciale *Mathematica* della Wolfram crea dei grafici piuttosto avanzati, e naturalmente il software si occupa anche di fare "i conti", ma *Mathematica*, ahimè, usa le proprie font che

$$\int_{-\infty}^{+\infty} e^{-t^2} dt = \sqrt{\pi}$$

FIGURA 3: integrale in LaTeXiano visto da *Mathematica*

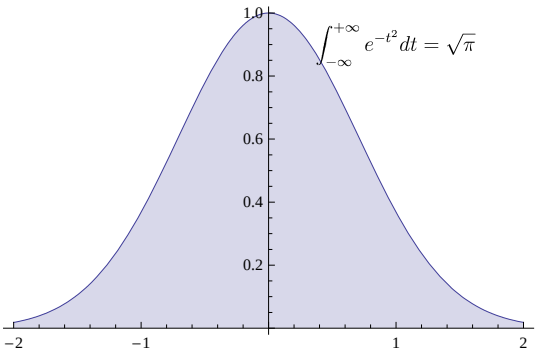


FIGURA 4: integrale e il suo grafico

sono diverse da quelle di TeX. Il lavoro è stato quello di implemetare un sistema per creare su *Mathematica* degli oggetti grafici che emulassero le font (e le formule) di LaTeX. Con Gnuplot è possibile fare una cosa molto simile, anche se in quel caso le formule LaTeX saranno vere font e non poligoni; inoltre sarà Gnuplot a creare il codice LaTeX del grafico e non viceversa. Il nostro approccio ha però il vantaggio che i poligoni della formula LaTeX si integrano nell'ambiente di *Mathematica*, quindi è possibile manipolarli come nell'esempio di figura 2 dove abbiamo inserito il grafico prodotto da `TeXClipping` in un grafico a tre dimensioni.

2 Installazione

È necessario aver installato i software gratuiti pstoedit e Ghostscript. A questo punto dopo aver caricato il pacchetto *Mathematica* dovreste già essere in grado di fare le prime prove a meno che un messaggio di errore vi avverta che, per qualche motivo, *Mathematica* non sia in grado di trovare gli eseguibili necessari (pstoedit o il compilatore TeX). In questo caso è necessario che individuiate manualmente il percorso e settiate l'ambiente con il comando `SetOptions[PstoeditPath->..,TeXPath->..]`. Naturalmente una volta individuati gli eseguibili necessari questo comando iniziale sarà sempre lo stesso. Infine, precisiamo che gli esempi sono stati fatti con la versione 6 di *Mathematica*, ma che il pacchetto funziona anche nella precedente versione 5.

3 Uso di base

Un esempio di come il pacchetto può essere usato, in maniera semplice per l'utente, è il seguente (vedi figura 1):

- Carico il pacchetto, se non è caricato in automatico.
- Scrivo

```
etichetta=TeXClipping[
"$$\int_{-\infty}^{+\infty}
{e^{-t^2}dt}=\sqrt\pi$$"
]
```

  nel foglio di lavoro di *Mathematica* (notebook) e ottengo la figura 3.
- Scrivo

```
Plot[Exp[-t^2], {t, -2, 2},
Filling -> Bottom,
Epilog -> Inset[etichetta, {1, .9}]]
```

  e ottengo la figura 4.

Tutto qui! Una sola funzione nuova: `TeXClipping`. Naturalmente ci sono alcune funzioni aggiuntive che permettono per esempio di scrivere le direttive precedenti con un unico comando `TeXInset` per ottenere direttamente la figura 4. Oppure alcune funzioni di debug: per esempio, se avessi dato il seguente comando: `TeXClipping["$\\beta"]` (nel qual caso si produce, ovviamente, un errore) allora è possibile saperne di più con `PrintError[]`, o cliccando sul tasto nel messaggio di errore, da cui otteniamo

```
Cannot compile the main.tex file!

\documentclass[12pt]{article}
\pagestyle{empty}\begin{document}
$\beta
\end{document}
:5:
Missing $ inserted
```

4 Le nuove funzioni

Le funzioni publiche sono le seguenti

- `TeXClipping[`*comTeX*`,`*opzioni*`]`: la funzione principale che trasforma la stringa *comTeX* di codice LaTeX in un grafico *Mathematica*. Con le *opzioni* `FontColor` e `Magnification` è possibile controllare il colore e le dimensioni della formula.
- `TeXInset[`*comTeX*`,`InsetArg`]`: funzione semplificatrice che, a partire dalla stringa *comTeX*, crea un grafico *Mathematica* `formLaTeX=TeXClipping[`*comTeX*`]` e restituisce un oggetto pronto per essere inserito in un grafico più grande.
- `PrintError[]`: Restituisce l'ultimo errore (utile sopratutto per fare il debug di una formula LaTeX).

- `GetFiles[`*dir*`]`: Per eventuali scopi di debug più approfonditi sposta i file ausiliari nella cartella *dir*.

5 Dettagli tecnici

Quello che abbiamo fatto è stato implementare una funzione di *Mathematica* che, prima con l'ausilio di pdfLATEX e poi con quello di pstoedit, trasformasse una stringa di codice TeX in un oggetto `Graphics` fatto di `Polygon`. Il primo passo è quello di individuare gli eseguibili necessari e creare una directory temporanea dove salvare i file ausiliari. Questo è ciò che fanno le seguenti funzioni

- `PrepareEnvironment`: la funzione cerca il compilatore LATEX, pstoedit e crea la cartella "TeXClipping" nella cartella temporanea di sistema.
- `GetPath[`*fileToFind*`]`: cerca il file *fileToFind*. Se il file è dato con il percorso completo restituisce semplicemente *fileToFind* altrimenti prima cerca il file di nome *fileToFind* nella cartella in cui si trova il nostro notebook `.nb` di lavoro corrente (il file sul quale stiamo lavorando); se non c'è, e solo in questo caso, lo cerca nella cartella del package `TeXClipping.nb` ed in entrambi i due casi ne restituisce il percorso completo. Se non lo trova affatto restituisce la stringa vuota.

I file ausiliari sono i seguenti

- `Main.tex` creato da *Mathematica* con all'interno il nostro codice LATEX.
- le funzioni usate sono `WriteTeXFileBody`, `WriteFile` e `WriteTeXFile`.
- `Main.pdf`, `Main.log`, `Main.aux` creati dal compilatore LATEX; e la funzione usata è `CompileTeX`.
- `pstoeditOutput.m` creato da pstoedit; e la funzione usata è `FormatPdf`.

Fin qui poche difficoltà, dato che finora il lavoro lo hanno fatto gli altri! Ora pstoedit, di Wolfgang Glunz, è un programma a linea di comando in grado di trasformare un file POSTSCRIPT o PDF in un'ampia gamma di altri formati vettoriali. È accessibile in maniera gratuita ed esporta in un `oggetto Graphics` di *Mathematica*. Precisamente, per i nostri scopi, pstoedit (con l'ausilio di Ghostscript) prende un file PDF creato da LATEX e ne restituisce un insieme di `Line`, linee che descrivono il contorno di ogni carattere. Purtroppo solo i contorni. Qui il nostro maggiore contributo: trasformare le linee in poligoni in modo che i caratteri risultassero anneriti nel loro interno. pstoedit restituisce un file di *Mathematica* pieno di linee e ora *Mathematica* deve interpretare le linee e formare i poligoni.

FIGURA 5: semplicemente "u" vista da *Mathematica*

FIGURA 6: la "o" sbagliata

5.1 Ottenere i poligoni

Trasformando ingenuamente un insieme di linee in un insieme di poligoni si può incorrere in qualche brutta sorpresa. Un semplice esempio chiarirà quale sia la problematica: durante l'esecuzione di `TeXClipping["u"]`, il pstoedit ci restituisce una linea che circonda la "u". Trasformando la linea in un poligono otteremo quello che vogliamo cioè una "u piena" come in figura 5. Ma se chiamiamo `TeXClipping["o"]`, da pstoedit otteremo due di linee, una per il cerchio più grande e una per quello più piccolo e trasformando tutto semplicemente in due poligoni otteremmo, visivamente, solo il cerchio grande annerito come in figura 6. Abbiamo peggiorato il risultato! Quello che dobbiamo fare è creare un unico poligono unendo i due cerchi. In questo modo la "o" si ottiene come una "u" dove avrei riunito le due estremità alte come in figura 7.

6 Dettagli ancora più tecnici (ma anche più interessanti)

Ci siamo imbattuti in alcuni problemi di topologia-geometria computazionale

6.1 La funzione `FillTeXSymbol`

La funzione `FillTeXSymbol` è la funzione *Mathematica* che si occupa di trasformare correttamente le linee ottenute da pstoedit in poligoni al fine di riempire i caratteri. Per prima cosa `FillTeXSymbol` trasforma l'insieme di liste di linee in un insieme di liste di poligoni X, come da manuale, poi applica ad X la regola `ruleForJoiningContainedPolygons` che individua le coppie di poligoni contenuti uno nell'altro e ne costruisce uno unico. Vediamo come avviene l'individuazione. Prendiamo una coppia di poligoni P_1 e P_2. Se chiamiamo R_1 ed R_2 i più piccoli rettangoli contenenti tutti i punti dei rispettivi poligoni P_1 e P_2, e risulta che R_1 è contenuto in R_2, allora P_1 potrebbe essere contenuto in P_2. In questo caso la funzione `rngMemberQ` restituirà vero se

FIGURA 7: semplicemente "o" vista da *Mathematica*

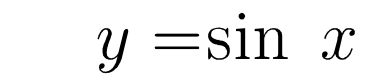


Figura 8: risultato di `Import`

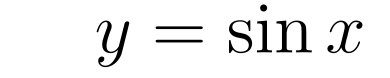


Figura 9: risultato di `TeXClipping`

due poligoni hanno passato il primo test, e quindi sono candidati ad essere uno contenuto nell'altro; per sapere se effettivamente un poligono circonda l'altro, useremo la circuitazione. La funzione `circuitation` implementa un integrale numerico lungo la "curva" formata dai punti del poligono e restituisce un valore prossimo a zero se il punto pt0 è circondato dalla curva pts.

7 Nota

La funzione `Import` di *Mathematica* versione 6 era stata la nostra prima scelta, ma purtroppo ha dei difetti che ci hanno costretti a indirizzarci a `pstoedit`. Immaginiamo di aver creato il file `main.pdf` compilando "$y = \sin x$". Se noi vogliamo ottenere un risultato simile con `Import["main.pdf"]` otteniamo il risultato in figura 8 mentre con `TeXClipping` otteniamo il risultato in figura 9! Nel primo caso il "sin" è spostato a sinistra verso l'uguale e ci sono alcune linee sottili che però si vedono in stampa.

▷ G. Gorni
`Gorni@dimi.uniud.it`

▷ S. Matiz
`matiz@dimi.uniud.it`

I font per le slide LaTeX resuscitati

Claudio Beccari

Sommario

La classe `slides` di LaTeX 2ε usava dei font molto leggibili ma con alcuni notevoli difetti quando si trattava di comporre la matematica. Da quando quella classe non è più curata dal LaTeX3 Team, quei font sono virtualmente scomparsi.

In questo articolo si recuperano quei font mostrando le correzioni eseguite e si illustra il pacchetto che ne consente l'uso anche con i programmi per produrre le belle presentazioni che con `slides` certamente non si potevano produrre.

Abstract

The LaTeX 2ε class `slides` used special fonts whose readability was exceptional; in spite of being part of the TeX system, they were not particularly efficient for what concerns mathematics. But since the time the LaTeX3 Team abandoned `slides` they almost disappeared.

This article describes the modifications and the enhancements made to these revived historical fonts and explains the package that make the new font version usable also with modern presentation classes so as to produce slides that were unthinkable of at the old times of `slides`.

1 Retrospettiva

Una volta esisteva il programma SliTeX Lamport, Leslie (1984), quando la versione dell'interprete TeX era ancora precedente alla 3.0. Erano gli anni '80, quasi la preistoria di TeX, sicuramente l'infanzia di LaTeX, quando per scrivere in una lingua diversa dall'inglese bisognava disporre di file di formato appositi, perché quel vecchio TeX non poteva gestire altro che un sistema di pattern di sillabazione alla volta.

Con la versione 3.0 l'uso di diverse lingue e di documenti scritti davvero facendo uso di diverse lingue ha permesso a TeX e a LaTeX di diffondersi ancor più nel vecchio continente, tanto che si è formato il LaTeX3 Team; come prima uscita, nel 1994, questa nuova squadra, tutta europea, ha prodotto LaTeX 2ε, e con questo è definitivamente morto SliTeX, sostituito dalla classe `slides` Lamport, Leslie (1994).

A parte il fatto di avere sempre lo stesso interprete e lo stesso linguaggio LaTeX per usare la classe `slides`, le cose non erano cambiate granché; i difetti che aveva SliTeX erano rimasti tutti; i difetti associati ai font erano rimasti invariati. In definitiva nel 1999 il LaTeX3 Team ha rinunciato alla manutenzione di questa classe, ma, anche se essa continua ad essere distribuita insieme ad ogni versione del sistema TeX, il Team ha formalmente invitato gli utenti di TeX a sviluppare nuove classi per creare delle presentazioni; questo caldo invito si rifletteva anche nel mettere a disposizione dell'intera comunità le loro creazioni.

Per questo siamo stati testimoni negli ultimi anni del proliferare di diversi sistemi per produrre bellissime presentazioni, colorate, animate, con font di ogni genere, ma sempre di tipo outline, adattissime alla proiezione con i moderni videoproiettori.

In questo articolo si descrivono i vecchi font usati da SliTeX e da `slides` mettendone in luce pregi e difetti; si illustrano le modifiche apportate, in particolare come si sia provveduto a generare i font PostScript di tipo 1, necessari per procedere al loro uso. Si descrive anche il file 'di stile' con il quale si possono sostituire tutti i font CM & AMS (oppure EC & AMS) per la produzione di belle presentazioni che usino questi font e sfruttino la loro grande leggibilità. Le soluzioni trovate possono essere considerate come una 'alpha-release'; se ci sarà abbastanza seguito si potranno riparare anche i difetti che eventualmente fossero ancora rimasti (e sicuramente ne sono rimasti tanti...).

2 I font per le citazioni

I font per le slide di SliTeX derivano dai font che Knuth ha creato per comporre le citazioni eleganti alla fine di tutti i capitoli dei suoi due libri fondamentali, *The TeXbook* Knuth, Donald E. (1990) e *The METAFONTbook* Knuth, Donald E. (2000).

Si tratta di font privi di grazie, formalmente di corpo 8, ma con un grande occhio e una sporgenza ridotta per ascendenti e discendenti. Chiunque abbia avuto per le mani quei libri sa perfettamente di che cosa si parli, ma qui viene riportato un semplice esempio, proprio per mostrare le successive modifiche.

> *If you can't solve a problem,*
> *you can always look up the answer.*
> *But please, Try first to solve it by yourself;*
> *then you'll learn more and you'll learn faster.*
>
> — DONALD E. KNUTH. *The TeXbook* (1983)

Si nota subito l'eleganza e lo stile di questo font lineare, semplice e perfettamente leggibile; benché esso sia in corpo 8, il suo occhio appare grande come quello del testo che lo precede, che è in corpo 10.

Il suo difetto, però, è che la 'L' minuscola, la 'i' maiuscola e, in matematica, il segno di 'modulo' sono indistinguibili. Questo font, elegante per le citazioni di Knuth, non è adatto alla composizione della matematica.

3 I font per le slide

Ecco perché, fin dai tempi di SliTeX, furono creati dei font speciali per le presentazioni, derivati da quelli per le citazioni, mediante, sostanzialmente, il cambio della 'i' maiuscola con una variante provvista di grazie.

If you can't solve a problem,
you can always look up the answer.
But please, Try first to solve it by yourself;
then you'll learn more and you'll learn faster.
— DONALD E. KNUTH. *The TeXbook* (1983)

4 I nuovi font per le slide

Come si vede, la 'i' maiuscola così modificata risolve in parte il problema, ma non completamente. Sarebbe opportuno modificare anche la 'L' minuscola in modo da eliminare anche l'ultimo residuo di possibile confusione con la matematica, con il segno di 'modulo' o di 'norma'.

Per questo sono stati creati dei nuovi font dove la 'L' minuscola è stata modificata disegnandola con un leggero arco al piede che sia in stile con il piede della 't' minuscola. Il risultato è il seguente.

If you can't solve a problem,
you can always look up the answer.
But please, Try first to solve it by yourself;
then you'll learn more and you'll learn faster.
— DONALD E. KNUTH. *The TeXbook* (1983)

Naturalmente questo ha portato con sé anche la modifica dei file metrici, ma più che altro la modifica di tutte le legature dove compare la 'l', sia le solite legature 'fl', 'ffl', ma anche tutte le lettere speciali con diacritici che compaiono con la codifica T1, oltre che nella solita codifica OT1 dove è coinvolta la lettera 'l'. Questo 'maquillage' ha coinvolto, naturalmente, anche le versioni nere, che dovevano necessariamente conformarsi alle versioni medie. Per completezza anche il Text Companion Font con encoding TS1 è stato corredato della famiglia sviluppata con i parametri di questo 'nuovo' font.

5 I problemi con la matematica

In matematica nascono altri problemi; nel vecchio SliTeX la matematica veniva composta con i normali font matematici usati anche fuori dall'ambito delle slide; il difetto che ne nasceva, a parte la disomogeneità di stile, era anche legato al fatto che il font per gli operatori veniva sostituito dai font tondi per le slide; questo di per sé introduceva una ulteriore disomogeneità, ma provocava anche la cattiva resa di quei segni matematici come per esempio le frecce 'lunghe', dove la freccia corta, resa con i normali font matematici, era prolungata da un segno 'meno' o da un segno 'uguale' tratto dai font per gli operatori. Il discorso sembra complicato, ma basta osservare $\Longrightarrow$ per rendersene conto; l'asse matematico è diverso; lo spessore delle linee è diverso; insomma si vede benissimo che si tratta di un risultato modestissimo, per non dire orrendo.

Per la matematica è stato quindi necessario creare dei font con gli stessi parametri usati per i font per le slide; questo ha comportato sia la creazione del font per le lettere (essenzialmente il corsivo matematico con una vasta collezione di simboli, comprese le lettere greche), del font per i simboli (quasi tutti i segni degli operatori, i numeri oldstyle e le lettere calligrafiche), il font per i delimitatori di dimensioni arbitrarie e gli operatori per le formule in display. La cosa non si è limitata a cambiare i parametri del master file in linguaggio METAFONT di ciascuno di questi font, ma di ritoccare ogni singolo segno, controllando l'adeguatezza alla perfetta composizione della matematica, se non altro per eliminare quegli obbrobri esemplificati sopra, ma anche per correggere altri piccoli dettagli al fine di portare l'intero set di font ad un insieme completo ed armonico dove ogni font è perfettamente adattato ad ogni altro font della serie.

6 Esempi

A parte l'ultimo esempio testuale riportato nel paragrafo 4 Si riportano ora alcuni esempi in cui si mescola testo e matematica.

6.1 Il neretto insieme al medio

Il titolino precedente è scritto con la versione ***nera*** del nuovo font, mentre questo testo è scritto con la versione ***media***.

Si noti che nell'esempio precedente, contrariamente a quello del paragrafo 4, la composizione è in corpo 10 come del resto anche il testo circostante, composto in corpo 10 ma con il font tondo della collezione Computer Modern; l'occhio più grande fa apparire l'esempio come se fosse composto almeno in corpo 12.

6.2 I corpi

Il nuovo font rispecchia il fatto che il font per le slide era e continua ad essere in un solo corpo, nel senso che il suo disegno è stato sviluppato per il corpo 8. Ogni altro corpo viene ottenuto per ingrandimento o per rimpicciolimento, come si vede dalla tabella 1.

Tabella 1 — Il nuovo font in vari corpi

Corpo	Esempio
5pt	ABCD abcd
7pt	ABCD abcd
10pt	ABCD abcd
12pt	ABCD abcd
14pt	ABCD abcd
17pt	ABCD abcd
20pt	ABCD abcd

L'esempio mostra chiaramente che i corpi più piccoli sono appena leggibili, mentre quelli più grandi appaiono in realtà un po' troppo neri. Ma questo è un difetto comune a tutti i font, compresi quelli PostScript, TrueType, OpenType, eccetera, che dispongono di una sola realizzazione, perfetta per un dato corpo, ma che degrada vistosamente quando la si ingrandisca o la si rimpicciolisca troppo.

In questo caso, scrivendo in corpo 10, i pedici e gli apici di primo livello appaiono ancora leggibili, ma non lo sarebbero i pedici e gli apici di secondo livello. Invece il corpo 20 appare più come un neretto, che come una serie media. In questo caso, però, i corpi grandi vengono usati prevalentemente per i titoli o i titolini, quindi non è un male se essi appaiono un poco più neri.

6.3 Confronto con il font lineare di default

È interessante anche il confronto a pari corpo con il font lineare della famiglia standard Computer Modern:

OT1/cmss	abcdefghijklmnopqrstuvwxyz
OT1/llcmss	abcdefghijklmnopqrstuvwxyz

e si può constatare come il nuovo font lineare appaia decisamente più grande rispetto al font lineare di default.

6.4 Un poco di matematica

Vale la pena di comporre qualche semplice espressione matematica.

L'equazione di secondo grado con coefficienti reali:

$$ax^2 + bx + c = 0 \tag{1}$$

ha le soluzioni

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{2}$$

con

$$\begin{cases} x_{1,2} \in \mathbb{R} & \text{se } b^2 - 4ac > 0 \\ x_1 = x_2 = -\dfrac{b}{2a} & \text{se } b^2 - 4ac = 0 \\ x_{1,2} \in \mathbb{C} & \text{se } b^2 - 4ac < 0 \end{cases} \tag{3}$$

Come si vede il corsivo matematico è decisamente corsivo e non può venire confuso con il corsivo del nuovo font, anche perché, come nei font lineari di default o anche in molti font commerciali, il carattere lineare non è dotato di corsivo vero e proprio, ma solo di un tondo inclinato. Nell'esempio appena riportato compaiono alcuni costrutti eseguiti con le potenzialità del pacchetto amsmath corredato anche delle estensioni dei font, tanto che sono stati usati anche i caratteri "black board bold", quelli ad aste raddoppiate, anche questi ricreati con i parametri grafici usati per il nuovo font lineare Mittelbach *et al.* (2004).

Le equazioni composte con i nuovi font sembrano stonare in un contesto in cui il font di default è il Computer Modern tondo con grazie; è meglio osservare l'effetto che essi fanno in una presentazione vera e propria, proiettata con il videoproiettore. La leggibilità sia del testo sia delle espressioni matematiche mantiene le promesse fatte dal vecchio SliTEX che puntava tutto sulla leggibilità.

6.5 Un altro poco di matematica

Non sarebbe possibile mostrare qui esempi dove vengano impegnati tutti i possibili segni standard di LATEX estesi con quelli del pacchetto amssymb. Tuttavia per valutare l'effetto ottenibile da questi font quando si componga la matematica vale la pena di esaminare qualche altra espressione, per esempio una espressione dove compaiano dei grandi operatori.

Il teorema dei residui afferma che se $f(z) : z, f \in \mathbb{C}$ è analitica in un dominio $\mathbb{D}$, salvo in un insieme finito di singolarità, allora vale la relazione

$$\oint_\gamma f(z)\,\mathrm{d}z = 2\pi\mathrm{j} \sum_{k=1}^{N_{\text{sing}}} R_k$$

dove γ è una linea chiusa singolarmente connessa completamente giacente in $\mathbb{D}$ e N_{sing} è il numero di singolarità contenute entro γ.

Si noti che nell'espressione appena presentata compare anche il simbolo del cerchio orientato in senso antiorario, tratto dalle polizze dei font della American Mathematical Society rivisti e ridisegnati per accordarsi con i font descritti qui.

Ora una formula con un integrale triplo e il font corsivo nero matematico ottenuto con il comando `\boldsymbol` del pacchetto amsmath.

$$\iiint_V F(\boldsymbol{P})\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z$$

Si potrebbe andare avanti ancora con altri esempi, ma si prolungherebbe inutilmente lo scritto, visto che tutte le strutture componibili con LATEX e i suoi pacchetti standard di estensione della matematica sono tutte compatibili con i nuovi font.

7 I simboli testuali

Siccome anche il Text Companion Font è stato ricomposto con i parametri di questa famiglia di font, tutti i simboli ottenibili con questa opzione sono perfettamente riutilizzabili insieme ai normali font per il testo e per la matematica. Qui compare un piccolo esempio dei simboli testuali producibili con questa varietà del Text Companion Font Mittelbach *et al.* (2004).

£ µ Ω ¥ $ € ℃

Si noti in particolare che il simbolo dell'euro, contrariamente al Companion non modificato, ma in conformità con la collezione di font lineari che stiamo trattando, è senza grazie.

Ovviamente anche questa varietà ha le forme e le serie del font testuale principale, vale a dire la serie nera sia per il tondo sia per il corsivo, e naturalmente il corsivo, che in realtà è solo il tondo inclinato, come avviene generalmente per quasi tutti i font lineari.

8 Trasformazione in font PostScript

Tutti i font descritti fin qui sono stati trasformati in formato PostScript binario facendo uso di mftrace Nienhuys, Han-Wen. Questo script Python può essere elaborato solo su macchine con sistema operativo di tipo UNIX; chi volesse usarlo su macchine Windows, deve prima installare l'ambiente (gratuito) di simulazione di UNIX, costituito dal programma Cygwin Autori vari (2007).

mftrace traccia i contorni dei font a matrici di pixel generati da METAFONT ricorrendo ad un altro programma, potrace Selinger, Peter (2007), che, tramite una serie di poligoni (da cui la parte iniziale del nome), determina i nodi e i punti guida delle spline di Bézier che permettono di descrivere i font PostScript. mftrace provvede ancora a far filtrare il risultato di potrace all'editor di font outline FontForge Williams, George (2007), in modo da ripulire il risultato da informazioni inutili o eccessive, e a formare i file di uscita nel formato giusto.

Per tutta questa operazione sono spesso necessari i vettori di codifica; per i font di estensione della American Mathematical Society questi vettori di codifica non erano disponibili nella distribuzione TeXlive, quindi si è provveduto a crearli; sono nel pacchetto di distribuzione nella stessa cartella dei font in formato binario e hanno estensione del nome `.enc`.

9 Installazione dei nuovi font

Per poter essere usati i nuovi font devono venire installati sul proprio sistema TeX.

Se ci si accontenta dei font a matrici di pixel, basta estrarre dal file compresso (con cui viene distribuito l'intero pacchetto contenente sia i font sia i file ausiliari) tutti i file con estensione `.mf` copinadoli poi in un file che si trovi in `$TEXTMFHOME/fonts/source/local/lxfonts`; se questa cartella manca, basta crearla. Il simbolo `$TEXTMFHOME` indica la radice dell'albero locale dove ogni utente conserva i propri file; per esempio, chi usasse MiKTeX su una macchina Windows potrebbe avere quel simbolo che corrisponde a `C:\localtexmf`; chi usasse Linux avrebbe quel simbolo che corrisponde a `~/texmf`; per gli utenti di Mac OS X esso corrisponderebbe a `~/Library/texmf`.

Dopo aver rigenerato il database dei nomi dei file (per Mac OS X non è necessario) ogni volta che serve un font a matrici di pixel ci pensa il sistema TeX a generare sia il file metrico `.tfm`, sia il file compresso con estensione `pk` che serve per visualizzare su schermo o per stampare su carta la propria composizione.

Siccome i file a matrici di pixel non sono particolarmente adatti a leggere i documenti direttamente sullo schermo, specialmente se si usano forti ingrandimenti, e non servono a nulla in quei sistemi in cui la compilazione del documento viene fatta direttamente in formato PDF[1], bisogna installare anche i file PostScript forniti con la distribuzione.

Questa è una operazione delicata e per i dettagli si rinvia alle istruzioni che accomapgnano il proprio sistema TeX, per esempio Lehman (2004). A grandi linee bisogna inserire i file `.pfb` della distribuzione in una cartella dal nome `$TEXMFHOME/fonts/type1/local/lxfonts`; bisogna copiare la mappa di questi font, per esempio in `$TEXMFHOME/fonts/map/dvips/lxfonts`[2]; bisogna aggiungere la riga

```
Map lxfonts.map
```

(rispettando la maiuscola) al file `updmap.cfg`; eseguire il programma `updmap` lanciandolo dalla finestra comandi, o console, o terminal, o xterm, comunque si chiami sul proprio calcolatore. Alla fine di queste poche operazioni, per altro insolite e un po' delicate, se non si sono commessi errori, i programmi del sistema TeX che possono usare direttamente i file PostScript risultano configurati anche per servirsi di questi nuovi font.

10 Il file di estensione per usare i nuovi font

Infine bisogna mettere il file `lxfonts.sty` nella cartella `$TEXMFHOME/tex/latex/lxfonts`, sempre ri-

1. In realtà il formato PDF dei documenti consente di usare font a matrici di pixel, ma quando si legge il documento direttamente sullo schermo, spesso questi caratteri presentano dei contorni mal definiti e a forti ingrandimenti presentano una sgradevole scalettatura dei contorni.

2. Nelle distribuzioni precedenti TeXlive 2007, la ricerca delle mappe veniva effettuata in una cartella diversa; in queste distribuzioni la mappa va copiata in `$TEXMFHOME/dvips/misc/lxfonts`

cordandosi di aggiornare il data base dei nomi dei file; ne sono esentati solo gli utenti di Mac OS X.

Il file `lxfonts.sty` è il punto di volta dell'intera configurazione per usare i nuovi font, ma va usato con cura.

È necessario fare attenzione a che esso sia caricato con `\usepackage` dopo aver caricato gli altri pacchetti per la gestione dei font tipici sia di LATEX 2_ε sia di `pdflatex`.

Se non si fanno scelte particolari per i font, questo pacchetto funziona con la codifica ridotta `OT1` (font con 128 caratteri e con le lettere accentate ottenute per sovrapposizione dell'accento), non carica la propria variante di Text Companion, e non carica le estensioni matematiche della American Mathematical Society.

Se invece si è soltanto specificato di usare per l'uscita la codifica `T1` mediante la richiesta

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

allora questo pacchetto si adegua automaticamente e carica i font con codifica estesa (font con 256 caratteri con le lettere accentate valide virtualmente per tutte le lingue).

Se si è richiesto il Text Companion Font, nuovamente, il pacchetto si adegua immediatamente e mette a disposizione del compositore tutti i vari comandi per introdurre nel testo i numerosissimi simboli che quel font mette a disposizione.

Se, infine, si è richiesto il pacchetto amsmath e, in particolare, si è specificato che si vogliono anche le estensioni dei font matematici mediante la richiesta

```
\usepackage{amsmath,amssymb}
```

allora il nuovo file lxfonts.sty immediatamente rende attivi i font contenenti le estensioni matematiche.

Non ci sono da installare i file di descrizione dei font, quelli con estensione `.fd`; tutto quello che LATEX ha bisogno di sapere a proposito di questi font è già contenuto dentro il file lxfonts.sty.

Chi avesse curiosità di leggere il contenuto di questo file troverebbe delle semplici dichiarazioni di famiglie e di forme di font che *per ogni corpo* richiedono di usare sempre gli stesi font, disegnati solo per il corpo 8.

Per questo motivo è particolarmente indicato fare uso della variante PostScript, e, anche se l'installazione e la configurazione sono un po' delicate, il gioco vale la candela.

11 Raccomandazioni

I piccoli esempi mostrati in questo testo 'cartaceo' mostrano pregi e difetti dei nuovi font, in particolare mettono in evidenza l'estrema leggerezza dei tratti dei corpi più piccoli, e questo è il maggiore se non l'unico difetto di questi font.

Usati invece come font per produrre presentazioni con beamer, per esempio, si ottiene il meglio, perché essi sembrano particolarmente indicati per le proiezioni (in fondo sono nati proprio per questo già con il vecchio SLITEX).

Per questo motivo mi sento di raccomandare di non usare questi font per testo corrente, ma solo per testo in grande corpo e per le presentazioni.

12 Conclusioni

Nel costruire questi font ho potuto constatare non pochi difetti nei file originari delle collezioni di segni messi a disposizione della American Mathematical Society; questi difetti non sono venuti in luce fino ad oggi, perché nessuno (presumo) ha avuto la necessità che ho avuto io di produrre nuovi font mediante gli stessi 'programmi' METAFONT, ma con parametri grafici assai diversi.

Lo stesso si può dire dei caratteri della collezione Text Companion, dove, per altro, ho dovuto (finora) correggere solo il simbolo dell'euro.

Sarebbero necessarie molte altre correzioni che solo l'uso, accompagnato da un notevole senso dell'osservazione, permettono di mettere in luce. La versione distribuita insieme a questo documento è quindi da ritenersi come una 'alpha-release', nemmeno ancora a livello 'beta'.

L'invito è quindi quello di provare ad usare questi nuovi font per le proprie presentazioni; ogni difetto venga segnalato così che possano venire apportate le correzioni giudicate indispensabili. Come con tutto il software libero, anche questi font si giovano del contributo critico degli utenti, oltre che del contributo creativo di chi ha la competenza per metter mano ai programmi METAFONT che descrivono ogni singolo segno.

Riferimenti bibliografici

Autori vari (2007). «GNU + Cignus + Windows = Cygwin». Il programma è installabile mediante un programma setup.exe da scaricare dal sito `http://www.cygwin.com/`.

Knuth, Donald E. (1990). *The TEXbook*. Addison Wesley, Reading Mass., 18ª edizione.

— (2000). *The METAFONTbook*. Addison Wesley, Reading Mass., 6ª edizione.

Lamport, Leslie (1984). *A document preparation system — LATEX — User's guide and reference manual*. Addison Wesley, Reading Mass., 1ª edizione.

— (1994). *A document preparation system — LATEX — User's guide and reference manual*. Addison Wesley, Reading Mass., 2ª edizione.

Lehman, P. (2004). «The font installation guide». Nella distribuzione di TEXlive 2007 questo documento è `TeXlive/`

2007/texmf-doc/doc/english/Type1fonts/fontinstallationguide.pdf.

Mittelbach, F., Goossens, M. *et al.* (2004). *The LaTeX companion*. Addison Wesley, 2ª edizione.

Nienhuys, Han-Wen. «mftrace — Scalable Fonts for METAFONT». Il programma è scaricabile da http://lilypond.org"/download"/sources"/mftrace"/mftrace-1.2.14.tar.gz.

Selinger, Peter (2007). «potrace — Transforming bitmaps into vector graphics». Il programma è scaricabile da http://potrace.sourceforge.net/.

Williams, George (2007). «FontForge». Il programma è scaricabile da http://fontforge.sourceforge.net/.

▷ Claudio Beccari
claudio dot beccari at alice.it

Utilizzo di caratteri TrueType con LaTeX
Un esempio pratico: i *Fell Types*

Massimiliano Dominici

Sommario

L'articolo mostra come è possibile sfruttare al meglio, con TeX, le potenzialità di un font TrueType. A questo scopo, viene esposta l'installazione di una collezione di font, i *Fell Types*, ricchi di caratteristiche non standard che TeX può essere istruito a gestire in maniera del tutto trasparente per l'utente.

Abstract

This paper explains how TeX can make the best use of the features of a TrueType font. For this purpose, the paper shows the installation of a collection of fonts, the *Fell Types*, full of non standard features that TeX can be taught to manage in a transperent way for the user.

1 Introduzione

Uno dei miti più duri da sfatare, riguardo a TeX, è quello secondo cui può essere utilizzato con un solo tipo di carattere[1] (il *Computer Modern*, naturalmente). Questa leggenda ha un suo fondo di verità. TeX ha potuto utilizzare, per molti anni, solo font in formato Metafont, dei quali non esisteva certo un'ampia scelta. E anche quando è stato possibile utilizzare font Type1 o TrueType, l'installazione di tipi di carattere diversi da quelli già presenti nella distribuzione usata ha continuato ad essere, a causa della complessità delle operazioni necessarie e della scarsa documentazione sull'argomento,[2] qualcosa che l'utente medio percepiva come al di sopra delle proprie capacità. Questa complessità, se da una parte implica anche un certo grado di complicazione, consente però di sfruttare le caratteristiche dei font ad un livello raggiunto dagli altri programmi di composizione tipografica solo con l'introduzione degli OpenType.

Oggi la diffusione sempre maggiore di quest'ultimo formato di font e lo sviluppo di estensioni di TeX (XeTeX e LuaTeX[3]) in grado di accedere direttamente ai font residenti nel sistema, senza dover passare attraverso i file ausiliari richiesti dall'implementazione di TeX, rende sicuramente più semplice l'utilizzo di nuovi tipi di carattere. Tuttavia, conoscere le potenzialità del vecchio sistema di accesso ai font (sistema che rimane comunque supportato anche dalle nuove implementazioni), consente di acquistare un controllo pieno sulle risorse disponibili, mettendo l'utente in grado di simulare le caratteristiche di un OpenType (sostituzione automatica di caratteri in base al contesto, ecc.) anche laddove queste non siano previste; oppure, nel caso si disponga di un OpenType, di "estendere" le funzionalità del font stesso al di là di quelle codificate al suo interno.

In questo articolo, dopo aver discusso, in generale, dell'installazione di un font TrueType, verrà mostrato un esempio concreto, scelto appositamente per la sua capacità di illustrare molte tra le manipolazioni possibili che TeX ci consente di operare su un font; e soprattutto quelle meno consuete. La specificità del font preso in esame fa sì che i metodi impiegati non possano essere generalizzati e applicati, così come sono, ad altri font, che, in genere, mostreranno diversi requisiti e diverse esigenze. Tuttavia tali metodi possono essere utili come "ricetta" da adattare alle circostanze.

Naturalmente, questo articolo non può esaurire l'intera materia relativa all'utilizzo dei font con LaTeX, di cui, invece, si limita ad evidenziare alcuni aspetti particolari. Per uno studio più sistematico dell'argomento rimandiamo, oltre che al classico e immancabile *TeXbook*,[4] al *LaTeX Companion*,[5] all'*Introduzione all'arte di scrivere con LaTeX* di Claudio Beccari,[6] e all'articolo di Emanuele Zannarini e Emiliano Giovanni Vavassori *LaTeX e i font: installazione pratica*,[7] che contiene una descrizione dettagliata dei vari formati di font esaminati nel presente articolo e un utilissimo glossario delle numerose estensioni di file riguardanti il vasto mondo dei caratteri digitali. Per una guida pratica all'installazione di font con LaTeX, infine, consigliamo la già citata *Font installation Guide*[8] di Philipp Lehmann.

1. Qui e di seguito uso il termine "carattere", o la locuzione "tipo di carattere", in riferimento ad un insieme di simboli per la stampa con caratteristiche grafiche comuni, considerate in maniera indipendente dalla tecnologia usata per tracciarli: il "tipo di carattere" *Times*. Il termine "font" si riferisce sempre, invece, alla sua versione digitalizzata.
2. La situazione è cambiata da quando è apparsa l'ottima guida pratica di Philipp Lehmann (Lehmann, 2004), ma la prima versione pubblica risale solo all'ottobre 2002.
3. Per maggiori informazioni su questi due programmi, si consultino i siti internet dei rispettivi progetti: `http://scripts.sil.org/xetex` e `http://www.luatex.org/`.
4. Knuth (1984).
5. Mittelbach *et al.* (2004).
6. Beccari (2007).
7. Zannarini e Vavassori (2005).
8. Lehmann (2004).

2 LaTeX e i caratteri da stampa

2.1 Lo schema di selezione dei font

Per determinare procedure e file ausiliari richiesti per poter utilizzare un font in generale (e un font TrueType in particolare), è necessario conoscere il modo in cui i font sono organizzati internamente, nello schema di selezione proprio di LaTeX 2$_\varepsilon$ ("New Font Selection Scheme", NFSS; si veda a questo proposito LaTeX3 Project Team (2005)).

LaTeX classifica ogni font a partire da cinque attributi: codifica, famiglia, serie, forma, corpo. Ogni volta che nel documento si ha un cambiamento di font (compresa la definizione del font iniziale) è come se venisse eseguita la seguente serie di istruzioni:

```
\fontencoding{<encoding>}
\fontfamily{<family>}
\fontseries{<series>}
\fontshape{<shape>}
\fontsize{<size>}{<baseline>}
\selectfont
```

L'argomento delle precedenti istruzioni viene immagazzinato nei rispettivi comandi interni: `\f@encoding`, `\f@family`, `\f@series`, `\f@shape`, `\f@size`, `\f@baselineskip`. Sono per l'appunto questi valori che vengono utilizzati per determinare il font da caricare. Per prima cosa, LaTeX esamina la combinazione codifica/famiglia, ovvero il valore di `\f@encoding/\f@family`. Se questo valore non è stato in precedenza dichiarato, LaTeX cerca un file corrispondente, con estensione `.fd`.[9] Ad esempio, nel caso in cui incontrasse, per la prima volta, la combinazione "T1/ptm", LaTeX andrebbe alla ricerca del file `t1ptm.fd`.

All'interno di questi file devono trovarsi le istruzioni per associare i valori contenuti nei comandi visti sopra con un preciso file `.tfm`, ovvero un file *TeX Font Metric*, che è tutto ciò che serve a LaTeX per comporre la pagina. L'effettiva inclusione del carattere, sia esso metafont, Type1 o TrueType, come vedremo in seguito, viene eseguita dal driver di stampa (sezione 2.2).

I "font definition files" contengono dunque delle tabelle che associano una determinata combinazione di codifica, famiglia, serie, forma e corpo ad un unico file `.tfm`. Queste tabelle vengono costruite per mezzo di due comandi:

```
\DeclareFontFamily{<encoding>}{<family>}
                  {<loading-settings>}
\DeclareFontShape{<encoding>}{<family>}
                 {<series>}{<shape>}
                 {<loading-info>}
                 {<loading-settings>}
```

`\DeclareFontFamily` oltre a dichiarare l'esistenza di una data famiglia per una data codifica, esegue il contenuto del terzo argomento ogni volta che viene caricato un font con tali caratteristiche. Ad esempio, per prevenire la divisione in sillabe dei caratteri a spaziatura fissa, è possibile passare come terzo argomento di `\DeclareFontFamily` il codice `\hypenchar\font=-1`. Il più delle volte, però, questo argomento è vuoto.

L'associazione effettiva di un file `.tfm` con un carattere avente specifici valori, viene effettuata tramite il comando `\DeclareFontShape`. In particolare in ⟨*loading-info*⟩ sono contenute le informazioni necessarie per associare i vari corpi con i font esterni. La sintassi è complessa e ci limiteremo qui ad esaminare un paio di esempi, in cui rimarcare i tratti essenziali. Il primo è tratto dal file `t1lmr.fd`, che contiene le definizioni per il carattere *Latin Modern Roman*, nella codifica T1. Quella che segue è la definizione inerente alla forma corsiva ("it") nel peso medio ("m"), ed è quella che si ottiene in questo documento[10] con il codice `\textit{Latin Modern Roman}`.

```
\DeclareFontShape{T1}{lmr}{m}{it}%
     {<-7.5>    ec-lmri7
      <7.5-8.5> ec-lmri8
      <8.5-9.5> ec-lmri9
      <9.5-11>  ec-lmri10
      <11->     ec-lmri12
      }{}
```

Le espressioni tra parentesi angolari rappresentano una gamma di dimensioni. Poiché il *Latin Modern* viene distribuito in diversi corpi, a differenti dimensioni vengono associati file differenti. Al di sopra di 11pt viene sempre utilizzato il file per il corpo 12, naturalmente adeguatamente scalato. Quando, poco sopra, LaTeX ha incontrato il codice `\textit{Latin Modern}`, ha consultato la dichiarazione precedente e ha trovato che per la combinazione "T1/lmr/m/it/10" doveva essere caricato il file `ec-lmri10.tfm`, dove si trovano le metriche richieste.

Il codice che segue è invece tratto da `t1yfrak.fd`, che contiene le definizioni per una versione dei caratteri gotici disegnati da Yannis Haralambous, in codifica T1.

```
\DeclareFontFamily{T1}{yfrak}{
   \hyphenchar \font =127}
\DeclareFontShape{T1}{yfrak}{m}{n}{
   <-> tfrak
}{}
\DeclareFontShape{T1}{yfrak}{m}{it}{
   <-> tfrakls
}{}
\DeclareFontShape{T1}{yfrak}{m}{sl}{
   <-> tswab
}{}
\DeclareFontShape{T1}{yfrak}{b}{n}{
   <-> tgoth
}{}
\DeclareFontShape{T1}{yfrak}{bx}{n}{
  <->ssub * yfrak/b/n
}{}
```

9. Sta per "font definition".

10. Questo documento, infatti, carica nel suo preambolo il pacchetto fontenc con opzione T1, per utilizzare la codifica T1 e il pacchetto lm, per utilizzare il carattere *Latin Modern*.

In questo caso i font sono distribuiti in un unico corpo e la cosa più rimarchevole è che si tratta di una collezione di font, più che una famiglia vera e propria. Infatti, il disegno di Fraktur (`tfrak`), Schwabacher (`tswab`) e Textur (`ygoth`) ha dei legami più tenui rispetto a quello dei nostri, rispettivi, tondo, corsivo e neretto.[11] Ciò mostra come sia possibile definire per LaTeX una famiglia di caratteri basata su font arbitrari.[12] Un'ultima notazione merita l'ultima dichiarazione del codice precedente. Indica la sostituzione di una serie ("bold-extended","bx") con un'altra ("bold","b"). LaTeX ha un meccanismo interno per gestire la sostituzione dei font mancanti. Senza entrare nei dettagli possiamo dire che nel caso in questione, senza l'ultima dichiarazione, ogni volta che avesse trovato un comando `\textbf` o `\bfseries`, mappati su "bx", LaTeX avrebbe fatto ricorso alla forma "m", come surrogato.

2.2 Inclusione dei font

A questo punto LaTeX sa quale file `.tfm` deve usare per comporre la pagina. Non gli serve altro. È come se mettesse in ordine, nelle posizioni appropriate, una serie di scatole vuote delle dimensioni giuste per accogliere i caratteri. L'inclusione vera e propria del font viene fatta da uno dei driver di stampa e la procedura varia a seconda del tipo di formato che si vuole ottenere. In realtà, il termine "driver di stampa" è un po' impreciso, perché in questo caso parliamo di un programma che genera un file (PDF, POSTSCRIPT, ecc.), oppure disegna i caratteri sullo schermo (nel caso di un visualizzatore di DVI, per esempio).

Il formato tradizionale di uscita di TeX è il DVI (DeVice Independent), pensato come formato intermedio che potesse essere trasformato nel linguaggio proprio di una qualunque stampante. Tuttavia, nel corso degli anni sono stati sviluppati diversi programmi per visualizzare direttamente a schermo i file in questo formato (Yap per Windows, Xdvi e Kdvi per sistemi *nix). Questi visualizzatori sono in grado di rendere direttamente dei font in formato TYPE1, oppure trasformare in un formato "bitmapped" (PK, Packed Font, estensione `.pk`), cioè non vettoriale,[13] i font in formato METAFONT e, con l'ausilio di `ttf2pk`, TRUETYPE.

Il meccanismo è pressappoco il seguente:

1. Il programma cerca nel file `psfonts.map` una riga che associ il `.tfm` usato ad un font TYPE1;
2. Se non lo trova cerca nelle appropriate cartelle un file `.pk` alla risoluzione richiesta oppure lo genera a partire da un file in formato METAFONT;
3. Se neanche questo è disponibile consulta la mappa `ttfonts.map` e, nel caso trovi una voce corrispondente, attiva `ttf2pk`.
4. Se non riesce a trovare un font da usare produce un messaggio di errore.

Lo stesso meccanismo è utilizzato da `dvips`, per generare un file POSTSCRIPT, che può eventualmente essere trasformato in PDF per mezzo dell'utilità `ps2pdf`, distribuita insieme all'interprete di POSTSCRIPT Ghostscript.

Per ottenere come formato finale un PDF, però, esistono soluzioni alternative, che permettono l'inclusione diretta di font di tipo TRUETYPE.[14] L'utilizzo, infatti, di font "bitmapped" sotto forma di TYPE3 in un documento PDF è sconsigliato, perché il font potrebbe apparire sgranato su alcuni visualizzatori, per quanto in fase di stampa non vi siano differenze apprezzabili.

La prima soluzione alternativa è quella di utilizzare `dvipdfm`, la cui procedura di inclusione dei font è simile, in pratica, a quella di `dvips`, con la differenza, appunto, che può includere direttamente font TRUETYPE, senza dover ricorrere ai `.pk`, e, di conseguenza, non effettua il terzo punto della procedura illustrata in precedenza (ed effettua le proprie ricerche nel file `dvipdfm.map`).

La soluzione più naturale, però, per ottenere un PDF è `pdftex`, a meno che non si abbiano vincoli particolari, come la presenza di grafici composti con PSTricks, ad esempio, che sfrutta caratteristiche avanzate del POSTSCRIPT non disponibili tramite `pdftex`. Di fatto, con installazioni di TeX ragionevolmente recenti, l'eseguibile `tex`, che si usa quando si vuole ottenere un DVI, è un collegamento proprio a `pdftex` in modalità DVI.

Con `pdftex` non è necessario, per ottenere un risultato visualizzabile, ricorrere ad una procedura in due stadi, come quella vista prima (composizione della pagina e poi, in un secondo tempo, inclusione dei caratteri tramite driver di stampa). È il programma stesso a provvedere all'inclusione dei font, cercando le associazioni tra `.tfm` e font veri e propri nella mappa `pdftex.map`. Anche in

11. Storicamente, però, il corsivo si è evoluto in maniera indipendente dal tondo, prima di diventare una variante all'interno della stessa famiglia.

12. Interviene, o dovrebbe intervenire, però, il gusto a porre dei limiti a tale arbitrarietà. Ciò non toglie che disponendo di un singolo tondo sprovvisto di un adeguato corsivo si può fare in modo che LaTeX usi automaticamente il corsivo di un'altra famiglia, se la sua forma si armonizza bene con quella del tondo.

13. I font in formato "bitmap" sono rappresentati tramite una matrice di punti (o mappa di bit) che indica al dispositivo incaricato di visualizzarli o di stamparli quale porzione dello schermo o della carta riempire di colore e quale no. Il risultato è che questo tipo di font richiede implementazioni diverse per dispositivi a diversa risoluzione. Un font "vettoriale", invece, dove i singoli caratteri vengono definiti tramite espressioni matematiche rappresentanti curve, può essere impiegato a risoluzioni diverse senza alcuna controindicazione.

14. Vedremo però che anche l'approccio descritto in precedenza consente, con alcuni accorgimenti, l'inclusione diretta di font TRUETYPE (sezione 4.5).

questo caso, essendo l'inclusione di un font TrueType diretta, il programma non ha bisogno di ricorrere a `ttf2pk`.

2.3 Mappe

Abbiamo visto nella sezione precedente che, per associare ad un determinato `.tfm` il relativo Type1 o TrueType, i vari programmi che si occupano dell'inclusione dei font fanno ricorso ad una o più mappe. Queste mappe possono essere installate nel sistema o caricate opzionalmente. In particolare `pdftex` in modalità PDF mette a disposizione la primitiva `\pdfmapfile` che può essere usata nel preambolo di un documento. Gli altri programmi, invece, come `dvips` o `dvipdfm`, essendo dei "post-processori", vengono chiamati da riga di comando ed è possibile, allora, caricare una particolare mappa passandola con l'appropriata opzione.

Tutte queste mappe, tranne quella utilizzata da `ttf2pk`, hanno una sintassi simile, anche se non identica, modellata su quella usata da `dvips`. In tutte, nessuna esclusa, deve essere indicato obbligatoriamente il file `.tfm`. Il nome del file contenente i caratteri può anche non essere specificato, nel caso che non si volesse includerlo, magari perché si vuole effettuare l'inclusione in un altro stadio del processo; in questo caso, però, deve essere indicato il nome "interno"[15] del font, in modo che il programma che, alla fine, dovrà fare l'inclusione, abbia un riferimento preciso.

In figura 1 è possibile esaminare la sintassi di questi file. È stata scelta a questo scopo la voce concernente il font LMRoman10-Regular, corrispondente al carattere *Latin Modern Roman* in corpo 10. Le righe relative a `psfonts.map` e `pdftex.map` sono identiche e mostrano nell'ordine: il nome del `.tfm`, il nome interno del font a cui si riferisce il `.tfm`, l'indicazione che il `.tfm` è stato ottenuto rimappando il file originale nella codifica (si veda, qui di seguito, la sezione 2.4) "enclmec", il file in cui tale codifica è specificata e, infine, il file in cui sono contenuti i caratteri. Le parentesi angolari che precedono il file di codifica e il Type1 indicano che tali file devono essere inclusi nell'output finale.[16]

La mappa usata da `dvipdfm` contiene invece un numero più stringato di informazioni, limitandosi al nome del `.tfm`, del vettore di codifica e del font.

Infine l'ultima voce è puramente ipotetica, dal momento che non è installato, di norma, nel sistema TeX un carattere *Latin Modern Roman* in formato TrueType. Si nota comunque che la sintassi è significativamente diversa da quella utilizzata per le altre mappe. Ricordiamo che questa mappa (`ttfonts.map`) viene utilizzata dai visualizzatori di DVI e da `dvips` per trasformare un TrueType in un font PK.

Sono state ovviamente trascurate molte delle opzioni che possono essere specificate in una mappa, e che indicano, ad esempio, le trasformazioni operate sul font per ottenere specifici effetti: se questo è stato inclinato, esteso, ecc. Per una visione più dettagliata della sintassi di questi file si possono consultare i manuali dei programmi `pdftex`, `dvips`, `dvipdfm` e `ttf2pk` inclusi nella propria distribuzione.

2.4 Vettori di codifica

Un vettore di codifica (in genere un file con estensione `.enc`) associa ad ogni singolo carattere contenuto in un dato font un valore numerico compreso tra 0 e 256. L'associazione avviene tramite il nome postscript del singolo carattere.[17] A tale nome corrisponde, nel font, una procedura che traccia il carattere in questione. In realtà, questo è strettamente vero solo per font di tipo POSTSCRIPT (Type1, Type3, ecc.), nei font TrueType i singoli caratteri sono identificati da un indice, invece che da un nome, ma, per questioni di compatibilità, esiste una tabella interna (`post`) che associa gli indici a nomi postscript.

All'interno di un file `.tfm` i singoli caratteri sono rappresentati appunto da valori numerici; il vettore di codifica traduce questi valori nei nomi delle procedure che il driver di stampa deve usare per tracciare, a schermo o sulla carta, i simboli corrispondenti.[18]

Ricodificare un font significa quindi associare agli stessi valori numerici caratteri (o meglio, simboli grafici) differenti. Questo può risultare utile (e lo è di fatto) per poter superare il limite di 256 caratteri accessibili da un singolo file `.tfm`.

2.5 Font virtuali

I font virtuali,[19] (*Virtual Fonts*, con estensione `.vf`) vengono menzionati qui per ultimi, ma rivestono un'importanza capitale se si vogliono sfruttare fino in fondo le potenzialità di un font. I font virtuali permettono, infatti, di operare sul font una serie di trasformazioni che vanno dalla ricodifica (si veda la sezione 2.4), alle alterazioni della forma (inclinazione, estensione, aggiunta di spazio addizionale, e in generale, qualsiasi trasformazione supportata dal linguaggio POSTSCRIPT), alla combinazione di caratteri provenienti da font reali differenti.

15. Il cosiddetto "nome postscript".

16. In particolare, la parentesi angolare singola posta davanti al Type1 significa che il font deve essere incluso solo parzialmente, cioè devono essere inclusi solo i caratteri effettivamente usati. Questo, oltre ad essere un modo per non appesantire eccessivamente il risultato finale, è spesso richiesto esplicitamente nella licenza dei font per evitare l'estrazione del font stesso dal documento. Due parentesi angolari avrebbero avuto il significato di inclusione completa.

17. È possibile vedere un esempio di vettore di codifica nella figura 2. L'associazione è implicita: al carattere in posizione 0, `/grave`, viene assegnato il valore numerico 0.

18. O, per usare un tecnicismo, i "glifi"; cioè i segni grafici che rappresentano concretamente i caratteri.

19. Per maggiori dettagli sull'argomento di questa sezione si veda Knuth (1990).

```
% Fonte: psfonts.map
ec-lmr10 LMRoman10-Regular "enclmec ReEncodeFont" <lm-ec.enc <lmr10.pfb

% Fonte: pdftex.map
ec-lmr10 LMRoman10-Regular "enclmec ReEncodeFont" <lm-ec.enc <lmr10.pfb

% Fonte: dvipdfm.map
ec-lmr10 lm-ec lmr10

% Fonte ipotetica: ttfonts.map
ec-lmr10 lmr10.ttf Encoding=lm-ec
```

FIGURA 1: Sintassi di un file `.map` per i diversi programmi.

Al suo interno un font virtuale indica i font reali da cui estrarre i singoli caratteri, i quali sono mappati su uno di questi (sotto forma di *TeX Font Metric* sprovvisto di *kerning* e legature; quello che comparirà nel file `.map`). Inoltre contiene tutte le informazioni (*kerning*, legature, trasformazioni su ogni singolo carattere) necessarie per rappresentare correttamente i caratteri.

TEX, però, non legge direttamente il font virtuale; abbiamo già visto che il suo compito consiste soltanto nel predisporre un insieme di scatole ben posizionate, pronte a ricevere, dal driver di stampa, i caratteri veri e propri. Quindi, le informazioni rilevanti per TEX (le metriche) sono duplicate in un file `.tfm`, compagno del font virtuale; ed è proprio questo file `.tfm` che viene letto da TEX in fase di composizione della pagina. Sarà il driver di stampa, leggendo nel font virtuale le trasformazioni da operare e l'associazione con i file `.tfm` "di base" (o "grezzi") rappresentati nelle mappe, a disegnare finalmente i caratteri nella forma voluta.

3 Installazione di un carattere TrueType

3.1 Panoramica generale

Ricapitolando quanto visto finora, ciò di cui abbiamo bisogno per poter utilizzare un font TRUETYPE con LATEX è:

1. Una serie di file *TEX Font Metric* `.tfm` che contengano le metriche del font (e, nel caso si vogliano sfruttare caratteristiche avanzate o effetti speciali, anche una serie di font virtuali `.vf`);

2. Una serie di file *Font Definition* `.fd` in cui ad una data combinazione di codifica/famiglia/serie/forma/corpo viene associato un particolare file `.tfm`;

3. Una serie di mappe in cui, ad un particolare file `.tfm` "di base" viene associato il corrispondente font;

4. Un vettore di codifica (`.enc`);

5. Infine un file di stile (`.sty`) per poter utilizzare convenientemente il font all'interno di un documento.

Le distribuzioni di LATEX attuali dispongono di una serie di utilità per gestire la creazione più o meno automatica di questi file. Le più comuni sono `ttf2tfm` e `fontinst`, che esamineremo nelle due sezioni seguenti. L'esame sarà, per forza di cose, sommario, e, per una conoscenza più dettagliata dei due programmi, si rimanda alla relativa documentazione.

3.2 Primo metodo: `ttf2tfm`

`ttf2tfm` viene distribuito insieme a `ttf2pk`. Si tratta di un programma da riga di comando con la seguente sintassi:

```
ttf2tfm ttffile[.ttf]
        [opzioni]
        [tfmfile[.tfm]]
```

Tramite le opzioni è possibile specificare la codifica del file "grezzo" e quella dell'eventuale font virtuale da generare, nonché applicare trasformazioni quali inclinazione o creazione di un finto maiuscoletto. L'esempio seguente servirà a dare un'idea più precisa del funzionamento:

```
ttf2tfm times.ttf -N -p 8r.enc \
-t T1-WGL4.enc -c .7 -V mtmrc8t.vpl \
-q mtmr8r.tfm >> ttfonts.map
```

In questo modo viene generato, a partire dal font TRUETYPE contenente Times New Roman, il file `.tfm` "grezzo" `mtmr8r.tfm`, nella codifica `8r`, nonché un file `.vpl`[20], da cui, mediante l'utilità `vptovf` è possibile ottenere il font virtuale `mtmrc8t.vf` e il suo compagno `mtmrc8t.tfm`. Questi ultimi rappresentano il maiuscoletto (finto) nella codifica `T1-WGL4`. L'opzione `-c` specifica di quanto vanno scalate le lettere maiuscole per ottenere il finto maiuscoletto, mentre l'opzione `-N` indica che, durante la ricodifica del font, bisogna accedere ai singoli caratteri mediante il loro nome postscript.

20. "Virtual Property List", è la versione in forma testuale dei file binari `.vf`.

Le informazioni normalmente mostrate sul terminale vengono redirette e aggiunte in coda al file `ttfonts.map`. L'opzione `-q` sopprime tutte le informazioni superflue, cosicché, alla fine, troveremo in fondo al suddetto file la seguente riga, che mappa il file `.tfm` "grezzo" `mtmr8r.tfm` sul file `times.ttf`:

```
mtmr8r times.ttf PS=Only Encoding=8r.enc
```

Il vettore di codifica "esterna", quello cioè utilizzato per creare il font virtuale, accetta una sintassi estesa rispetto a quella di un normale file `.enc`, che contiene solo una lista ordinata di 256 nomi di carattere. Le informazioni opzionali servono, ad esempio, ad istruire `ttf2tfm` su come comporre le legature richieste, oppure a sopprimere del *kerning*. In figura 2 è mostrato un estratto dal file `T1-WGL4.enc`. Da riga 27 a riga 31 si possono osservare le istruzioni per definire le legature; da riga 48 a riga 53 e poi da riga 124 a riga 131, rispettivamente, l'inizio e la fine della codifica vera e propria.

Un limite di `ttf2tfm` è che non è in grado di combinare insieme caratteri provenienti da font diversi. Questo, naturalmente, è un limite molto relativo per un programma pensato esplicitamente per operare su font TrueType in cui, di norma, tutti i caratteri, anche quelli appartenenti al cosidetto *expert set* (legature inusuali, maiuscoletto, numeri minuscoli, ecc.) quando sono presenti, si trovano inclusi nello stesso file. Tuttavia, questo malauguratamente non è sempre vero, e nell'esempio presentato più avanti (sezione 4) vedremo che saremo costretti ad assemblare caratteri da file diversi.

Un'altro limite è che il file di mappa ha una sintassi, come abbiamo visto anche in precedenza (si veda la figura 1), molto diversa da quella delle altre mappe e difficilmente traducibile (ad esempio non c'è modo di recuperare il nome postscript del file). Né `ttf2tfm` mette a disposizione alcuno strumento per generare mappe in un formato diverso.

3.3 Secondo metodo: `fontinst`

`fontinst` è un programma, completamente scritto in TeX, in grado di leggere un file di testo contenente istruzioni metriche in differenti formati (di norma un file Adobe Font Metric è un buon punto di partenza) e tradurre, dopo averle eventualmente modificate, queste istruzioni nel linguaggio tipico delle (*Virtual*) *Property List*. I file di testo `.pl` e `.vpl` creati da `fontinst` possono poi essere trasformati in `.tfm` e `.vf` per mezzo dei due programmi `pltotf` e `vptovf`, disponibili in ogni distribuzione di TeX. `fontinst`, inoltre, provvede alla creazione automatica dei file *Font Definition* (sezione 2.1) e semiautomatica delle mappe (sezione 2.3).

Un file `.tfm` "grezzo" può essere ottenuto, con `fontinst`, passando l'istruzione

\transformfont{*<tfm>*}{*<transforms>*}

L'argomento *<tfm>* è il nome del file da generare; all'interno dell'argomento *<transforms>* vanno indicate le operazioni da effettuare sul file stesso. Una delle più comuni è la ricodifica:

\reencodefont{*<encoding>*}{\fromafm{*<afm>*}}

Abbiamo specificato `\fromafm{`*`<afm>`*`}`, perché, il più delle volte si parte da un file `.afm`, ma altre istruzioni lecite avrebbero potuto essere `\frommtx{`*`<mtx>`*`}` o `\frompl{`*`<pl>`*`}`.

Un font virtuale può essere invece ottenuto con l'istruzione `\installfont`:

\installfont{*<vf>*}{*<mtx>*}{*<etx>*}
{*<encoding>*}{*<family>*}{*<series>*}
{*<shape>*}{*<size>*}

Il primo argomento è il nome del file `.vf`; gli argomenti da *<encoding>* in poi sono gli attributi del file, come verranno registrati nel file `.fd`. Le metriche e la codifica del file sono specificati, tramite appositi file ausiliari generati al volo oppure provvisti dal sistema o dall'utente, negli argomenti *<mtx>* e *<etx>*.

In particolare *<mtx>* provvede alla parte metrica vera e propria, specificando il *kerning* estratto dall'AFM e indicando quali caratteri è possibile simulare a partire da quelli già esistenti, e come, e quali no. Il *kerning* si trova di solito in un file con estensione `.mtx`, avente lo stesso nome di un `.tfm` "grezzo", e generato al volo da `fontinst` a partire da questo. La costruzione dei caratteri è invece contenuta in un file (sempre con estensione `.mtx`) provvisto dal sistema e dipendente dall'alfabeto usato. Nel caso dell'alfabeto latino si tratta di `newlatin.mtx`. L'istruzione `\unfakable{`*`<glyph>`*`}` indica che il carattere *<glyph>* è disponibile solo se esiste esplicitamente all'interno del font. In alcuni casi è possibile ottenere un carattere a partire da altri già esistenti. È il caso della legatura `fi` (l'esempio è tratto da `newlatin.mtx`):

```
\setglyph{fi}
   \glyph{f}{1000}
   \movert{\kerning{f}{i}}
   \glyph{i}{1000}
\endsetglyph
\setleftkerning{fi}{f}{1000}
\setrightkerning{fi}{i}{1000}
```

Il significato di questo codice è: "se non è disponibile un carattere `fi` distinto, usa il carattere `f`, poi spostati di uno spazio uguale al *kerning* tra `f` e `i`, quindi usa il carattere `i`; per finire, attribuisci a questo carattere a sinistra il *kerning* che avrebbe il carattere `f` e a destra quello che avrebbe il carattere `i`".

All'interno di un file `.etx`, da utilizzare nell'argomento *<etx>* di `\installfont`, si trova invece specificata la codifica del font virtuale. Ad ogni

```
 1  % T1-WGL4.enc
 2  %
 3  %
 4  % This is LaTeX T1 encoding for WGL4 encoded TrueType fonts
 5  % (e.g. from Windows 95)
27  % LIGKERN f i =: fi ;
28  % LIGKERN f l =: fl ;
29  % LIGKERN f f =: ff ;
30  % LIGKERN ff i =: ffi ;
31  % LIGKERN ff l =: ffl ;
48  /T1Encoding [          % now 256 chars follow
49  % 0x00
50    /grave /acute /circumflex /tilde
51    /dieresis /hungarumlaut /ring /caron
52    /breve /macron /dotaccent /cedilla
53    /ogonek /quotesinglbase /guilsinglleft /guilsinglright
124 % 0xF0
125   /eth /ntilde /ograve /oacute
126   /ocircumflex /otilde /odieresis /oe
127   /oslash /ugrave /uacute /ucircumflex
128   /udieresis /yacute /thorn /germandbls
129 ] def
130
131 % eof
```

Figura 2: Estratti dal vettore di codifica `T1-WGL4.enc`

carattere è assegnata una posizione e sono specificate inoltre le legature, sotto forma di tabella di sostituzioni. Quello che segue è un esempio tratto da `t1.etx`:

```
\setslot{\lclig{FF}{ff}}
  \ifnumber{\int{ligaturing}}>{0}\then
    \ligature{LIG}{%
      \lc{I}{i}}{\lclig{FFI}{ffi}
     }
    \ligature{LIG}{%
      \lc{L}{l}}{\lclig{FFL}{ffl}
     }
  \Fi
  \comment{The 'ff' ligature. It should
    be two characters wide in a monowidth
    font.}
\endsetslot
```

L'istruzione `\setslot` attribuisce al carattere `ff` (`\lclig` sta per "lowercase ligature", legatura minuscola, ed è definita in modo da assumere il valore del suo secondo argomento) un indice numerico, tramite un contatore progressivo, che ne stabilisce la posizione all'interno del vettore di codifica. Le istruzioni seguenti stabiliscono che il carattere in questione può formare una legatura se seguito dai caratteri `i` o `l`, (e specificatamente le legature `ffi` e `ffl`), ma solo se la variabile `ligaturing` è maggiore di zero. Questa possibilità di avere alcune caratteristiche legate ad un'espressione condizionale, rende estremamente semplice generare, a partire da uno stesso file `.etx`, diversi font virtuali che sfruttano diverse risorse del font reale.

Per ricapitolare, l'equivalente di quanto visto nella sezione 3.2 (creazione di un font virtuale contenente un falso maiuscoletto di Times New Roman, nella codifica T1) si può ottenere, quindi, scrivendo le seguenti righe in un file di testo e lanciando `tex` sul file stesso:

```
\input fontinst.sty
\setint{smallcapsscale}{700}
\recordtransforms{mtm-rec.tex}
\transformfont{mtmr8r}
  {\reencodefont{8r}{\fromafm{mtmr8a}}}
\installfonts
\installfamily{T1}{mtm}{}
\installfont{mtmrc8t}{mtmr8r,newlatin}{t1c}
  {T1}{mtm}{m}{sc}{}
\endinstallfonts
\endrecordtransforms
\bye
```

Alla fine della compilazione avremo un file *Property List* (`mtmr8r.pl`) da cui ricavare il corrispondente *TeX Metric File* "grezzo"; un font *Virtual Property list*, `mtmrc8t.vpl`, da cui ricavare il *Virtual Font* e il *TeX Font Metric* suo compagno; il file `t1mtm.fd`, con le associazioni tra gli attributi del font e i corrispettivi file metrici e il file `mtm-rec.tex` contenente le informazioni necessarie per generare mappe in vari formati.

Il seguente codice, una volta compilato con `tex`, permette di generare mappe per `pdftex`, `dvips` e `dvipdfm`:

```
\input finstmsc.sty
\adddriver{dvips}{mtm.map}
\adddriver{dvipdfm}{dvipdfm-mtm.map}
\input mtm-rec.tex
\donedrivers
\bye
```

Un'ultima notazione, infine. Tutto il procedimento seguito finora presuppone, al primo anello della catena, l'esistenza di un file AFM. I font TrueType, però, ne sono sprovvisti, in quanto, a differenza dei font Type1, le metriche sono immagazzinate

all'interno del font stesso, in apposite tabelle. Per "estrarre", quindi, un AFM da un TrueType, è necessario utilizzare il programma `ttf2afm`, che fa parte di una normale distribuzione di TeX, come nel seguente esempio:

```
ttf2afm -e 8a.enc -o mtm8a.afm times.ttf
```

È necessario specificare una codifica (opzione `-e`), altrimenti tutti i caratteri vengono mappati nella posizione `-1`, e l'AFM così generato è praticamente inservibile.

4 I *Fell Types*

4.1 I caratteri

I *Fell Types* sono una collezione di font digitalizzati da Igino Marini (`http://www.iginomarini.com/fell.html`), che riproducono alcuni tra i caratteri acquistati o commissionati alla fine del XVII secolo dal Dr John Fell per la tipografia dell'università di Oxford. La collezione comprende otto tipi di carattere, il cui nome indica, secondo le convenzioni dell'epoca, le dimensioni del corpo (si veda la tabella 1). Cinque di essi, utilizzabili per il corpo del testo, sono composti da tondo, corsivo e maiuscoletto; uno, per la titolazione, è privo delle minuscole e presenta come unica forma il tondo; e gli ultimi due offrono un centinaio scarso di ornamenti tipografici.

Il disegno dei caratteri, opera in gran parte dell'incisore olandese Peter de Walpergen, con il forte contrasto fra tratti fini e spessi e la relativa brevità degli ascendenti, è tipico della scuola olandese dell'epoca ed è uno dei primi esempi di transizionale barocco. Il carattere *English*, invece, dal disegno più raffinato (non a caso è stato preso a modello per le altre digitalizzazioni dei Fell Types, operate da *Hoefler & Frères* e *Dutch Type Library*) mostra connessioni più marcate con la tradizione francese e potrebbe essere opera di Christoffel Van Dijck, per quanto riguarda il tondo, e di Robert Granjon per il corsivo.

I caratteri originali comprendono, oltre ai caratteri usuali, alcune legature non consuete (`ct`, `st`, `fr`, ecc.), una variante della "s" minuscola (`longs`) e le legature di questa con altri caratteri, nonché varianti in posizione terminale o iniziale di alcune lettere. I numerali appaiono solo nella forma minuscola.

In fase di digitalizzazione sono stati aggiunti alcuni caratteri accentati e, soprattutto, un *kerning* molto accurato, ottenuto con il programma *iKern*, realizzato dallo stesso Igino Marini. Nessun font supera la soglia di 256 caratteri rappresentabili in un singolo vettore di codifica.

I *Fell Types* possono essere usati liberamente per la composizione di documenti, alla sola condizione di riprodurre nel documento stesso la nota di attribuzione "The Fell Types are digitally reproduced by Igino Marini. www.iginomarini.com"[21]. Attualmente, ad esempio, sono utilizzati nell'ambito del progetto *Electronic Texts in American Studies.*[22]

4.2 Preparazione dell'installazione

Prima di cominciare l'installazione dei caratteri[23], bisogna avere ben chiaro in mente il tipo di utilizzo che se ne intende fare. Possiamo pensare di usare i caratteri separatamente oppure come un'unica collezione, associando i singoli font a diverse dimensioni del corpo. Inoltre possiamo definire un'interfaccia che ci permetta di attivare a richiesta, come se avessimo a disposizione delle tabelle OpenType, caratteristiche particolari. Per esempio: attivare solo le legature tradizionali; attivare tutte le legature, più la "s lunga"; distinguere tra "s lunga" (all'inizio o all'interno di una parola) e "s" normale (alla fine di una parola).

L'interfaccia potrebbe essere ancora più complessa. Si potrebbe aggiungere l'uso automatico delle forme varianti. Si potrebbe scorporare l'uso della "s lunga" dall'uso delle legature, rendendola una caratteristica ortogonale alle altre. Tuttavia, questo, oltre ad aumentare il livello di complicazione, moltiplicherebbe a dismisura il numero dei font virtuali e metrici necessari a rappresentare il sistema. Ci limiteremo, quindi, ad un'interfaccia relativamente più semplice. I risultati che si ottengono attivando uno dei tre livelli di legature sono mostrati in figura 3, applicati alla prima frase dell'opera *Pseudodoxia Epidemica, or Vulgar Errors* di Thomas Browne, scrittore inglese della metà del XVII secolo.

The First and Father-cause of common Error, is, The common infirmity of Human Nature.

The Firſt and Father-cauſe of common Error, if, The common infirmity of Human Nature.

The First and Father-cauſe of common Error, is, The common infirmity of Human Nature.

FIGURA 3: Diversi livelli di legatura

Dal momento che sarà necessario avere un numero consistente di file *Font Definition*, relativi ai singoli tipi di carattere e alla collezione nel suo insieme, e di mappe, per i vari programmi; e che dovremo assemblare in un unico font virtuale caratteri provenienti da font diversi, la scelta del programma da utilizzare per l'installazione cade naturalmente su `fontinst`.

21. Le condizioni per la distribuzione sono più restrittive. Si possono consultare i termini della licenza sul sito dell'autore.

22. `http://digitalcommons.unl.edu/etas/`.

23. Il codice da cui sono tratti gli esempi di questa sezione e delle seguenti è consultabile per intero all'indirizzo `http://www.guit.sssup.it/downloads/felltypes.tar.gz`.

Nome	Incisore	Tondo	Corsivo	Maiuscoletto	Ornamenti	Corpo (pt)
Pica	De Walpergen	Sì	Sì	Sì	No	10.5
English	Van Dijck (?) – Granjon (?)	Sì	Sì	Sì	No	11.5
Great Primer	De Walpergen	Sì	Sì	Sì	No	14
Double Pica	De Walpergen	Sì	Sì	Sì	No	17
French Canon	De Walpergen	Sì	Sì	Sì	No	33
Three Line Pica	De Walpergen	Sì	No	No	No	41
Flowers1	Granjon et al.	No	No	No	Sì	25
Flowers2	Granjon et al.	No	No	No	Sì	17.5

TABELLA 1: I *Fell Types*

4.3 File ausiliari

Come abbiamo visto in precedenza (sezione 3.3) dobbiamo, per prima cosa, generare gli AFM richiesti da `fontinst`. Per fare questo, però, abbiamo bisogno di un vettore di codifica che non ci faccia perdere per strada qualche carattere.[24] L'ideale sarebbe usare, per ogni font, la codifica interna del font stesso. Come è possibile procurarsela, possibilmente con un procedimento automatizzabile? Un primo metodo consiste nel lanciare l'eseguibile `ttf2afm` sul font in questione, senza specificare il vettore di codifica. Nell'AFM compaiono, tutti mappati in posizione `-1`, i caratteri nell'ordine in cui vengono trovati. L'AFM può quindi essere modificato con un programma di manipolazione del testo (come `awk`, ad esempio, se si dispone di un sistema *nix), per ottenere il vettore di codifica nel formato voluto.

Alternativamente, si può fare ricorso a FONTTOOLS, un modulo Python reperibile all'indirizzo `http://sourceforge.net/projects/fonttools/`, che permette, tra le altre cose, di accedere ad una lista dei caratteri nella codifica propria del font. Basandosi su questo modulo e utilizzando il linguaggio Python, è possibile, appunto, trasformare questa lista in un file `.enc` sintatticamente corretto.

Successivamente, lanciando `ttf2afm` su ciascun file con l'appropriato vettore di codifica, si ottengono gli AFM voluti. Queste operazioni, che andrebbero ripetute su diciotto font, possono essere automatizzate abbastanza facilmente tramite uno script.

È invece, purtroppo, impossibile rendere automatica la creazione dei file `.mtx` e `.etx`. È possibile, però, utilizzare come base i corrispettivi file distribuiti con `fontinst`: `newlatin.mtx` e `t1.etx`.

Il file `newlatin.mtx` è strutturato in maniera modulare: le definizioni relative ai singoli caratteri sono distribuite nei file `llbuild.mtx`, `lubuild.mtx`, `ltpunct.mtx`, `lsfake.mtx`, `lsbuild.mtx`, `lsmisc.mtx`. È possibile, quindi, riutilizzare il codice contenuto nei singoli moduli, includendoli, mediante gli appropriati comandi, nel file `fell.mtx`, che potrà poi essere utilizzato nello script di `fontinst`. Non è necessario includere gli ultimi tre moduli sopra menzionati, in quanto essi contengono codice relativo al trattamento di font cosiddetti CAPS & SMALL CAPS, che usano per i caratteri in maiuscoletto nomi postscript come `Asmall`, ecc. Questo non avviene con i *Fell Types*, dove questi caratteri hanno gli stessi nomi dell'alfabeto minuscolo. Le definizioni relative ai caratteri non standard presenti nei *Fell Types* possono essere inserite, sotto forma di istruzioni `\setglyph` (vedi sezione 3.3), all'interno di un ulteriore modulo (`fell-specific.mtx`) da aggiungere agli altri. Si è scelto di utilizzare `\setglyph` piuttosto che `\unfakable` anche con caratteri non composti, in modo da avere un default cui ricorrere in mancanza di meglio. Così sarà possibile, utilizzando ad esempio l'*English*, usare il carattere `ealt` sapendo che, se si decide di ricomporre il documento in *Pica*, che non possiede tale carattere, in quel punto apparirà una `e` e non un quadratino nero.

Per quanto riguarda il file `.etx` (`t1fell.etx`), si può ottenere editando una copia di `t1.etx`, eliminando i riferimenti a caratteri che non compaiono nei *Fell Types* e sostituendoli con altri, oppure con *slot* vuoti. Può essere interessante esaminare il codice relativo alla lettera `s`, perché è un buon esempio di come si può controllare l'attivazione, in un font virtuale, di una data caratteristica.

```
\ifnumber{\int{ligaturing}}<{1}\then
\setslot{s}
  \comment{The ‘s’ character}
\endsetslot
\Else
\setslot{longs}
  \ligature{LIG}{h}{longsh}
  \ligature{LIG}{i}{longsi}
  \ligature{LIG}{l}{longsl}
  \ligature{LIG}{longs}{dbllongs}
  \ligature{LIG}{t}{longst}
  \ifnumber{\int{ligaturing}}>{1}\then
  \ligature{LIG}{boundary}{s}
  \ligature{LIG/}{quoteright}{s}
  \ligature{LIG/}{guilsinglleft}{s}
  \ligature{LIG/}{guilsinglright}{s}
  \ligature{LIG/}{quotedblleft}{s}
  \ligature{LIG/}{quotedblright}{s}
  \ligature{LIG/}{guillemotleft}{s}
  \ligature{LIG/}{guillemotright}{s}
  \ligature{LIG/}{period}{s}
  \ligature{LIG/}{comma}{s}
  \ligature{LIG/}{colon}{s}
```

24. Si richiama l'attenzione del lettore sul fatto, già accennato nella sezione 3.3, che ogni carattere presente nel font ma non contemplato dal vettore di codifica viene mappato in posizione `-1` e viene di conseguenza ignorato dai programmi che leggono l'AFM.

```
  \ligature{LIG/}{semicolon}{s}
  \ligature{LIG/}{rangedash}{s}
  \ligature{LIG/}{punctdash}{s}
  \ligature{LIG/}{exclam}{s}
  \ligature{LIG/}{question}{s}
  \ligature{LIG/}{slash}{s}
  \ligature{LIG/}{hyphen}{s}
  \ligature{LIG/}{parenleft}{s}
  \ligature{LIG/}{parenright}{s}
  \Fi
  \comment{The 'longs' character.}
\endsetslot
\Fi
```

Se la variabile `ligaturing` ha valore minore di 1, allora viene usata la "s" normale, altrimenti viene usata la variante "lunga" con tutte le sue legature. Ma, se la variabile è maggiore di 1 la "s lunga" viene di nuovo trasformata in "s" normale in fine di parola (ovvero quando è seguita da uno spazio bianco, `boundary`, o da punteggiatura). Il codice `LIG/` indica un tipo particolare di legatura, dove il secondo carattere che va a formare la legatura viene mantenuto. Più oltre, nello stesso file, ci si assicura che anche la legatura `longst` si comporti alla stessa maniera di `longs` e si trasformi in `st` in fine della parola.

Infine, poiché è conveniente costruire font virtuali e metrici a partire da una versione "ricodificata" del font, è necessario anche preparare due vettori di codifica che contengano, rispettivamente, tutti i caratteri presenti nella forma tondo e corsivo l'uno (`fell.enc`), e maiuscoletto l'altro (`fell-sc.enc`).[25]

4.4 File metrici e definizioni

Una volta concluse queste operazioni preliminari, è possibile preparare lo script che istruirà `fontinst` nel processo di creazione dei file metrici. Contestualmente verranno creati anche i file *Font Definition* e le mappe.

```
\input finstmsc.sty
\input fontinst.sty
\needsfontinstversion{1.927}
\enctoetx{fell}{fell}
\enctoetx{fell-sc}{fell-sc}
\recordtransforms{fell-rec.tex}
```

Dopo aver caricato `finstmsc.sty` e `fontinst.sty`, e aver dichiarato la versione di `fontinst` in uso, i due vettori di codifica `fell.enc` e `fell-sc.enc` vengono trasformati in file `.etx` tramite `\enctoetx`. Quindi viene attivata la registrazione delle operazioni sul file `fell-rec.tex`. Sarà a partire dalle informazioni trascritte automaticamente su questo file che verranno, in seguito, generate le mappe per i vari programmi.

I file metrici "grezzi" ricodificati si ottengono, come è stato mostrato in sezione 3.3, tramite il comando `\transformfont`. Nel codice seguente, tale operazione viene mostrata limitatamente al tondo, corsivo e maiuscoletto del tipo di carattere *Pica*.

```
\transformfont{picar}{%
  \reencodefont{fell}{\fromafm{FePIrm2}}}
\transformfont{picari}{%
  \reencodefont{fell}{\fromafm{FePIit2}}}
\transformfont{picarc}{%
  \reencodefont{fell-sc}{\fromafm{FePIsc2}}}
```

Generati i file metrici "grezzi", è necessario trasformarli in font virtuali e creare i *Font Definition* corrispondenti. Questo avviene all'interno della coppia di comandi `\installfonts`, `\endinstallfonts`. In precedenza, però, va attivato il livello di legatura voluto, con il comando `\setint{ligaturing}{<int>}`

```
\setint{ligaturing}{0}
\installfonts
\installfamily{T1}{fell0}{}
\installfont{ecanonr0}
  {ecanonr,ecanonrc,fell}{t1fell}
  {T1}{fell0}{m}{n}{<-10.95>}
\installfont{picar0}
  {picar,picarc,fell}{t1fell}
  {T1}{fell0}{m}{n}{<10.95-14.4>}
\installfont{greatpr0}
  {greatpr,greatprc,fell}{t1fell}
  {T1}{fell0}{m}{n}{<14.4-17.28>}
\installfont{dpicar0}
  {dpicar,dpicarc,fell}{t1fell}
  {T1}{fell0}{m}{n}{<17.28-24.88>}
\installfont{fcanonr0}
  {fcanonr,fcanonrc,fell}{t1fell}
  {T1}{fell0}{m}{n}{<24.88->}
\installfont{ecanonri0}
  {ecanonri,ecanonrc,fell}{t1fell}
  {T1}{fell0}{m}{it}{<-10.95>}
\installfont{picari0}
  {picari,picarc,fell}{t1fell}
  {T1}{fell0}{m}{it}{<10.95-14.4>}
\installfont{greatpri0}
  {greatpri,greatprc,fell}{t1fell}
  {T1}{fell0}{m}{it}{<14.4-17.28>}
\installfont{dpicari0}
  {dpicari,dpicarc,fell}{t1fell}
  {T1}{fell0}{m}{it}{<17.28-24.88>}
\installfont{fcanonri0}
  {fcanonri,fcanonrc,fell}{t1fell}
  {T1}{fell0}{m}{it}{<24.88->}
\installfont{ecanonrc0}
  {ecanonrc,fell}{t1fell}
  {T1}{fell0}{m}{sc}{<-10.95>}
\installfont{picarc0}
  {picarc,fell}{t1fell}
  {T1}{fell0}{m}{sc}{<10.95-14.4>}
\installfont{greatprc0}
  {greatprc,fell}{t1fell}
  {T1}{fell0}{m}{sc}{<14.4-17.28>}
\installfont{dpicarc0}
  {dpicarc,fell}{t1fell}
  {T1}{fell0}{m}{sc}{<17.28-24.88>}
\installfont{fcanonrc0}
  {fcanonrc,fell}{t1fell}
  {T1}{fell0}{m}{sc}{<24.88->}
\endinstallfonts
```

Il codice mostrato qui sopra illustra, per l'appunto, l'installazione della famiglia di font `fell0`, corrispondente all'intera collezione di font contenente

25. `fontinst` richiederebbe a questo scopo un file `.etx`, ma poiché è in grado di costruirselo da solo sulla base di un `.enc`, e quest'ultimo è molto più semplice da scrivere, il procedimento sopra indicato non ha conroindicazioni.

soltanto le legature usuali (variabile `ligaturing` uguale a 0). Il comando `\installfamily` è responsabile della scrittura, nel file `t1fell0.fd`, della riga `\DeclareFontFamily{T1}{fell0}{}` (si veda la sezione 2.1), mentre ogni chiamata a `\installfont` genera un file `.vpl` e contemporaneamente crea una voce nel file `t1fell0.fd` che associa il file metrico ad una determinata combinazione di attributi.

Il codice riguardante ogni singola istruzione `\installfont` è stato diviso su tre righe sia per esigenze di spazio, sia per separare anche visivamente i diversi compiti svolti dall'istruzione stessa. Nella prima riga è indicato il nome del file `.vpl` (e quindi del font virtuale e del *TeX Font Metric* che ne risulteranno), ad esempio `picar0`. Nella seconda sono indicati i file ausiliari necessari per generare il file `.vpl`, ovvero i file `.mtx` `picar`, `picarc` e `fell`, e il file `.etx` `t1fell`. Poiché all'interno del tondo e del corsivo mancano alcuni caratteri presenti nella codifica (ad esempio le parentesi graffe), è necessario recuperarle dal corrispondente maiuscoletto: da qui l'esigenza di usare `picarc` accanto a `picar`. Infine nella terza riga sono indicate le informazioni che verranno scritte nel file `t1fell0.fd` sotto forma di istruzioni `\DeclareFontShape`. Si può notare che ad ogni *TeX Font Metric* è associata soltanto una gamma di dimensioni, ad esempio `picar0` verrà usato solo per dimensioni che vanno da 10.95pt a 14.4pt.

Dal momento però che vogliamo poter usare i vari tipi di carattere anche singolarmente, è necessario installare le singole famiglie. Non è, tuttavia, necessario creare nuovi `.vpl`. Il codice che segue, e che va ripetuto per ogni famiglia di font, si limita a creare il file `t1pica0.fd`, associando le diverse combinazioni di attributi al *TeX Font Metric* indicato nel primo argomaento di `\installfontas`.

```
\installfonts
\installfamily{T1}{pica0}{}
\installfontas{picar0}{T1}{pica0}{m}{n}{}
\installfontas{picari0}{T1}{pica0}{m}{it}{}
\installfontas{picarc0}{T1}{pica0}{m}{sc}{}
\endinstallfonts
```

Queste operazioni vanno ripetute immutate, cambiando la variabile `ligaturing` e ponendola uguale prima a 1 e poi a 2, in modo da avere le famiglie di font `t1fell1` e `t1fell2`, `t1pica1` e `t1pica2`, ecc., relative agli ulteriori due livelli di legature.

Lo stesso procedimento è valido per installare le medesime famiglie di font nella codifica TS1, *Text Companion*, che contiene i simboli utilizzati in modalità non matematica. Una tipica istruzione `\installfont`, in questo caso, ha il seguente aspetto:

```
\installfont{tc-picar0}
  {picar,picarc,textcomp}{ts1}
  {TS1}{fell0}{m}{n}{<10.95-14.4>}
```

L'unica differenza rispetto all'installazione della famiglia in codifica T1, è rappresentata dall'uso del file `textcomp.mtx` al posto di `fell.mtx` e di `ts1.etx` al posto di `t1fell.etx`. Va sottolineato il fatto che non è necessario generare i font virtuali in codifica TS1 anche per il maiuscoletto, né per ogni livello di legature; basterà quindi utilizzare istruzioni `\installfontas` invece che `\installfont`.

Restano da installare ancora il tipo di carattere *Three Line Pica*, usato per la titolazione, e le due collezioni di ornamenti. Questi ultimi, essendo font contenenti caratteri non testuali, non hanno bisogno di ricorrere ad una codifica. Per indicare questo fatto, verrà usata la pseudo codifica U (*Unknown*).

```
\installfonts
\installfamily{U}{flow}{}
\installrawfont{FeFlow1}
  {FeFlow1}{txtfdmns,FeFlow1 mtxasetx}
  {U}{flow}{m}{n}{}
\installrawfont{FeFlow2}
  {FeFlow2}{txtfdmns,FeFlow2 mtxasetx}
  {U}{flow}{m}{sc}{}
\endinstallfonts
```

Inoltre, come si vede dal codice qui sopra, verrà usato direttamente il file `.tfm` "grezzo" (istruzione `\installrawfont`). Non essendo specificata alcuna codifica si indica a `fontinst` di dedurla dal file `.mtx` stesso (`mtxasetx`), aggiungendo, ad ogni buon conto, i parametri `\fontdimen` che servono a TEX per poter utilizzare il font (`txtfdmns`).

Three Line Pica, invece, pur essendo un tipo di carattere per il testo, deve essere utilizzato solo in particolari contesti, come titoli, capilettera, ecc. Inoltre non è influenzato in alcuna maniera dalla variabile `ligaturing`. Ciò significa, una volta generato il corrispondente file metrico "grezzo", l'installazione è semplicissima:

```
\installfonts
\installfamily{T1}{tlpica}{}
\installfont{tlpicar}
  {tlpicaraw,fell}
  {t1fell}{T1}{tlpica}{m}{n}{}
\endinstallfonts
```

Lo script si conclude chiudendo il file `fell-rec.tex` e uscendo dal programma.

```
\endrecordtransforms
\bye
```

Alla fine della compilazione con `tex`, saranno presenti nella cartella di installazione un certo numero di file con estensione `.pl` e `.vpl`. Lanciando su questi file, rispettivamente, gli eseguibili `pltotf` e `vptovf`, si otterranno i font virtuali e i *TeX Font Metric*, da copiare nelle cartelle `fonts/vf/imfell/` e `fonts/tfm/imfell/`. I file *Font Definition* e i TrueType possono essere spostati, così come sono nelle cartelle `tex/latex/imfell/`

e `fonts/truetype/imfell/`.[26] Tutte queste operazioni possono essere automatizzate nello script di installazione.

4.5 Mappe

Come illustrato in precedenza (sezione 3.3), `fontinst` è in grado di generare mappe per i più comuni driver di stampa (`dvips`, `pdftex` e `dvipdfm`), con un procedimento in due fasi. Nella prima fase lo script utilizzato per installare i font genera un file in cui sono raccolte tutte le informazioni rilevanti. Nella seconda, questo file viene richiamato all'interno di un secondo script che si occupa specificamente di generare le mappe.

Sfortunatamente, se il font da installare è un TrueType, le cose si complicano. Innanzitutto deve essere generata anche una mappa per `ttf2pk`, in modo che i visualizzatori di DVI possano correttamente disegnare i font a schermo. In secondo luogo nelle mappe per `dvips` e `pdftex` deve essere indicato un vettore di codifica anche quando il generatore di mappe, pensato per font Type1, originalmente incluso in `fontinst` non lo farebbe (cioè quando il font non è stato ricodificato). Questo perché altrimenti `pdftex` non è in grado di effettuare l'inclusione parziale dei font.

Infine va considerato un ultimo fatto. `dvips` non è in grado di includere un font TrueType in un documento PostScript al volo sotto forma di font Type42.[27] Quindi, un documento realizzato con `dvips` contiene solo font bitmap generati a partire dai TrueType. Questo, specialmente se il documento deve essere ulteriormente distillato in PDF, può non essere un risultato auspicabile. Si può, allora, far ricorso a Ghostscript, il quale, invece, è in grado di fare l'inclusione al volo, sia per visualizzare il documento con GSView o altri visualizzatori, sia per distillare in PDF con `ps2pdf`. Per fare questo, però, è necessario che nella mappa per `dvips` non vengano specificati i font da includere, dimodoché il programma inserisce solo un riferimento e sarà poi l'inteprete di PostScript, cioè, nel caso in esame, Ghostscript, ad effettuare materialmente l'inclusione. Tutto questo significa che è necessario preparare due diverse mappe per `dvips` e `pdftex`.

Bisogna quindi scrivere, in un file che verrà chiamato `adddrivers.sty`, dei nuovi generatori di mappe, usando come base quelli forniti da `fontinst` nel file `finstmsc.sty` e modificandoli in modo da ottenere i risultati voluti. I nuovi generatori, a cui è bene attribuire un nome che ricordi che sono stati creati appositamente per gestire file TrueType, verranno poi richiamati all'interno dello script `fell-map.tex`, come si vede nel codice che segue.

```
\input finstmsc.sty
\input adddrivers.sty
\resetstr{PSfontsuffix}{.ttf}
\adddriver{ttfdvips}{dvips-fell.map}
\adddriver{ttfpdftex}{pdftex-fell.map}
\adddriver{ttfdvipdfm}{dvipdfm-fell.map}
\adddriver{ttftopk}{ttf2pk-fell.map}
\input fell-rec.tex
\donedrivers
\bye
```

Inoltre la variabile `PSfontsuffix` va posta uguale a `.ttf`, altrimenti `fontinst` utilizzerebbe il suffisso di default, che è `.pfa`.[28]

Le mappe così ottenute andranno copiate nelle cartelle dove possono essere trovate dai rispettivi programmi. Nello specifico: `fonts/map/dvips/`, `fonts/map/pdftex/`, `fonts/map/dvipdfm/` e `fonts/map/ttf2pk/`. Di norma, nel caso di un font Type1 la mappa è solo una, e il suo contenuto può essere automaticamente aggiunto, con le dovute modifiche, alle mappe globali lette dai vari programmi. Di solito esiste, nella propria distribuzione di TeX, una utilità (che nella TeX Live si chiama `updmap`) che si occupa proprio di questo.

Nel nostro caso questo non è possibile e ogni mappa va trattata separatamente. Se la compilazione viene effettuata tramite `pdftex` la mappa può essere caricata direttamente dal documento grazie alla primitiva `\pdfmapfile`. Se la compilazione è avvenuta con `tex` il visualizzatore DVI e `dvips` non trovando nessuna voce relativa ai *TeX Font Metric* all'interno della mappa globale cercheranno automaticamente in `ttfonts.map`, dove si sarà provveduto ad aggiungere il contenuto di `ttf2pk.map` (manualmente o tramite comando nello script di installazione). Infine, se si vuole ottenere un PDF con `dvipdfm` o con `dvips` seguito da `ps2pdf`, è necessario passare la mappa al compilatore tramite l'apposita opzione (`-f` per `dvipdfm` e `-u` per `dvips`).

4.6 File di stile

È consigliabile avere a disposizione una serie di istruzioni per non dover ricorrere a comandi di basso livello ogni volta che si vuole cambiare il tipo di carattere usato, o il suo livello di legature. Queste istruzioni vanno inserite in un apposito file di stile da caricare nel documento, con il consueto comando `\usepackage`. Qui di seguito non verrà illustrato l'intero contenuto del file di stile, ma solo alcuni tratti salienti. Si rimanda, come al solito, ai commenti del sorgente, per una spiegazione completa.

26. Si intende, qui e oltre, sempre a partire dalla radice `$TEXMF/`

27. Un font Type42 è sostanzialmente un font TrueType inglobato in un contenitore compatibile con il linguaggio PostScript.

28. Non entreremo qui nei dettagli di come scrivere un generatore di mappe. Come al solito, il lettore curioso può consultare il codice a disposizione all'indirizzo `http://www.guit.sssup.it/download/`.

Nel file di stile in questione dovrà essere possibile specificare il tipo di carattere e il livello di legature usati, sia come opzione del pacchetto, sia in un qualsiasi punto del documento. Il pacchetto kvoptions offre una serie di strumenti (ricalcati su quelli usati internamente in pacchetti come hyperref o geometry) per specificare le opzioni del pacchetto nella forma "chiave = valore". Verrà perciò caricato nel preambolo del nostro file di stile.

```
\ProvidesPackage{imfell}
\RequirePackage{graphics}
\RequirePackage{kvoptions}
\SetupKeyvalOptions{%
  family=IM,
  prefix=IM@,
}
```

Inoltre, se il documento viene compilato con `pdflatex`, deve essere caricata l'apposita mappa, per cui bisognerà ricorrere all'espressione condizionale `\ifpdf`.

```
\RequirePackage{ifpdf}
\ifpdf
  \pdfmapfile{+pdftex-fell.map}
\fi
```

Il punto fondamentale è che il nome delle famiglie dei font da utilizzare è costituito dal nome del font vero e proprio (o della collezione: `fell`) seguito da un numero arabo che indica il livello di legature. Il modo più semplice per specificare separatamente questi due elementi è dichiarare le variabili che le contengono, `\IM@basefamily` e `\IM@liglevel`, e utilizzare poi queste all'interno dei comandi di basso livello. Così, per passare da una famiglia all'altra e da un livello di legature all'altro si possono utilizzare i seguenti comandi:

```
\DeclareRobustCommand{\switchtoliglevel}[1]{%
  \edef\IM@tempa{#1}
  \def\IM@liglevel{\IM@tempa}%
  \fontfamily{\IM@basefamily\IM@liglevel}%
  \selectfont%
}
\DeclareRobustCommand{%
  \switchtofamily}[2][\IM@liglevel]{%
    \edef\IM@tempa{#1}
    \switchtoliglevel{#1}%
    \def\IM@basefamily{#2}%
    \fontfamily{\IM@basefamily\IM@liglevel}%
    \selectfont%
}
```

Per quanto riguarda il comando `\switchtofamily` è possibile specificare anche il livello di legature come argomento opzionale. Oltre a questi comandi generali, possono apparirne anche altri più specifici, costruiti su di essi.

```
\DeclareRobustCommand{\textpi}[2][\IM@liglevel]{%
  {\switchtofamily[#1]{pica} #2}%
}
```

Il comando `\textpi` serve a comporre il testo racchiuso nell'argomento obbligatorio, nel tipo di carattere *Pica*, specificando eventualmente anche il livello di legature desiderato.

Infine, è utile avere un comando per richiamare, in maniera semplice, un determinato ornamento; qualcosa del tipo `\fellornament{31}`, dato che è più comodo identificare un ornamento con un numero che con un nome. Per ottenere qualcosa di simile è necessario prima catalogare tutti gli ornamenti e, di conseguenza, definire un comando che faccia appunto questo. Il codice che segue è una proposta in tal senso.

```
\DeclareRobustCommand{\fellornament}[1]{%
  \csname flow@orn@#1\endcsname
}
\newcommand{\DeclareTextOrnament}[7]{%
  \expandafter \def\csname #1@orn@#2\endcsname{%
    \usefont{#3}{#4}{#5}{#6} \char#7%
    }%
}
```

A questo punto, dopo aver inserito nel file di stile la riga

```
\DeclareTextOrnament{flow}{31}{U}{flow}{m}{sc}{1}
```

il comando `\fellornament{31}` inserisce nel documento l'ornamento che si trova nella posizione "1" del font `FeFlow2`. Infatti alla forma `sc` corrisponde, secondo quanto è scritto nel file `uflow.fd`, proprio quel font (sezione 4.4).

Giunti alla conclusione di questa complessa e laboriosa installazione, i *Fell Types* sono finalmente in grado di essere utilizzati in un documento che ne sfrutti appieno le particolarità.

4.7 Un esempio

Qui di seguito (figura 4) è mostrata una pagina che riproduce l'inizio del primo capitolo dell'opera, citata nella sezione 4.2, *Pseudodoxia Epidemica, or Vulgar Errors* di Thomas Browne. Non verrà qui riprodotto il codice[29] che ne è alla base, in quanto, per la maggior parte, quel codice non ha niente a che vedere con i *Fell Types*; ma ci si limiterà ad un paio di considerazioni che sono attinenti al presente lavoro.

Innanzitutto, l'utilizzo dei tipi di carattere, in questo caso si tratta dell'intera collezione con tutte le legature attivate, è tanto semplice quanto inserire le seguenti righe:

```
\usepackage[T1]{fontenc}
\usepackage[default,liglevel=2]{imfell}
```

Il corpo del testo è composto in *Pica*, con le note a margine in *English*. Il titolo della sezione è in *Great Primer*, con l'intestazione in *Double Pica*. Il titolo del capitolo, infine è in *French Canon* e *Great Primer*. Per il capolettera è stato utilizzato il *Three Line Pica*.

29. Come al solito, però esso è consultabile nell'archivio `felltypes.tar.gz`, reperibile all'indirizzo `http://www.guit.sssup.it/download/`

THE

FIRST BOOK:

OR

GENERAL PART

CHAP. I

Of the Cauſes of Common Errors.

Introduction

THE First and Father-cauſe of common Error, is, The common infirmity of Human Nature; of whoſe deceptible condition, although perhaps there ſhould not need any other eviction, than the frequent Errors we ſhall our ſelves commit, even in the express declarement hereof: yet ſhall we illuſtrate the ſame from more infallible conſtitutions, and perſons preſumed as far from us in condition, as time, that is, our first and ingenerated forefathers. From whom as we derive our Being, and the ſeveral wounds of conſtitution; ſo, may we in ſome manner excuſe our infirmities in the depravity of thoſe parts, whoſe Traductions were pure in them, and their Originals but once removed from God. Who notwithſtanding (if posterity may take leave to judg of the fact, as they are aſſured to ſuffer in the puniſhment) were groſly deceived, in their perfection; and ſo weakly deluded in the clarity of their underſtanding, that it hath left no ſmall obſcurity in ours, How error ſhould gain upon them.

Matter of great diſpute, how our fathers could be ſo deceived

For first, They were deceived by Satan; and that not in an inviſible inſinuation, but an open and diſcoverable apparition, that is, in the form of a Serpent; whereby although there were many occaſions of ſuſpition, and ſuch as could not eaſily eſcape a weaker circumſpection, yet did the unwary apprehenſion of *Eve* take no advantage thereof. It hath therefore ſeemed ſtrange unto ſome, ſhe ſhould be deluded by a Serpent, or ſubject her reaſon to a beast, which God had ſubjected unto hers. It hath empuzzled the enquiries of others to apprehend, and enforced them unto ſtrange conceptions, to make out, how without fear or doubt ſhe could diſcourſe with ſuch a creature, or hear a Serpent ſpeak, without ſuſpition of Impoſture. The wits of others have been ſo bold, as to accuſe her ſimplicity, in receiving his Temptation ſo coldly; and when ſuch ſpecious effects of the fruit were Promiſed, as to make them like God; not to deſire, at least not to wonder he purſued not that benefit himſelf. And had it been their own caſe, would perhaps have replied. If the tast of this Fruit maketh the eaters like *Gods*, why remainest

FIGURA 4: Capitolo iniziale di *Pseudodoxia Epidemica, or Vulgar Errors*, composto con i *FellTypes*

I disegni ornamentali sono stati ottenuti componendo insieme uno o più ornamenti provenienti dai due font *Flowers*. Ciò può essere fatto utilizzando, a basso livello, la primitiva di TEX `\xleaders`, che consente di ripetere un carattere (o meglio: una scatola che può contenere anche più di un carattere) fino a riempire completamente la dimensione specificata. Il problema è che, se tale dimensione non è coperta da un numero intero di elementi, `\xleaders` lascia ovviamente degli spazi bianchi di uguali dimensioni tra i vari blocchi. Questo problema è stato risolto, dato che l'elemento ornamentale può essere considerato un elemento grafico non assoggettabile ai vincoli della griglia tipografica, disponendo i suddetti blocchi su una dimensione il più possibile vicina a quella voluta e riscalando poi il risultato. In questo modo, è facilissimo inserire i due elementi in questione inserendo rispettivamente le due righe di codice che seguono.

```
\OrnamentRule{11}{2}{3}{12}{\textwidth}
\OrnamentRule{51}{51}{51}{51}{.5\textwidth}
```

Il codice per `\OrnamentRule` è incluso nel file `imfell.sty` e può qui essere consultato.

5 Conclusioni

5.1 Problemi residui

La presenza (e l'uso) di caratteri non standard nei *Fell Types* rende problematica l'estrazione di testo da un PDF che includa tali font. Infatti l'applicazione che visualizza il documento deve essere istruita ad interpretare tali caratteri come una sequenza di uno o più caratteri standard (ad esempio: `ct` → `c t`). Questo è possibile se l'applicazione trova, all'interno del documento, una tabella che, per ogni carattere non standard, associa all'indice del carattere il valore Unicode dei suoi componenti. Purtroppo solo il percorso di compilazione `tex` → `dvips` → `ps2pdf`, è in grado di inserire nel PDF una mappa /ToUnicode, e anche in questo caso limitatamente a caratteri presenti nello standard Unicode[30] (ad esempio `longs`, `longst`, `ct`, ma non `longsh` e `ll`).

`pdftex` ha un paio di primitive non documentate, `\pdfgentounicode` e `\pdfglyphtounicode`, che permettono di generare la mappa /ToUnicode e di personalizzarla, aggiungendo specifiche per caratteri non contenuti nello standard Unicode. Sfortunatamente queste primitive funzionano solo con font TYPE1.

5.2 Risultati

Il percorso affrontato in questo lavoro ha mostrato come è possibile, con TEX, sfruttare, in una maniera semplice per l'utente finale, tutte le caratteristiche di un font.

Partendo da una collezione di font ricca di caratteri inconsueti (i quali, insieme al *kerning* accurato, rappresentano la caratteristica saliente dei *Fell Types*) e difficilmente accessibili dall'interno di un programma di videoscrittura, in assenza di tabelle di sostituzione come quelle OPENTYPE, siamo stati in grado di offrire all'utente finale un'interfaccia minimale, non intrusiva, e tuttavia capace di fornirgli tutti gli strumenti necessari per gestire la ricchezza e la complessità di questi font.

Certamente tutto questo viene poi scontato con il dover affrontare un processo di installazione che non può assolutamente essere definito semplice, né può essere riportato a procedure standard da applicare ciecamente, almeno in casi complessi come quello appena visto. Tuttavia anche questa considerazione negativa è attenuata dal fatto che, grazie al carattere aperto di TEX, una volta prodotta una soluzione, questa viene di solito messa a disposizione della comunità che può utilizzarla, studiarla e migliorarla, soprattutto, senza che ogni utente debba ripercorrere ogni volta la stessa strada.

30. Le legature, e altre "forme di presentazione" non sono considerate caratteri; quelle presenti nello standard unicode lo sono solo per motivi di compatibilità. Si veda a questo proposito `http://www.unicode.org/faq/ligature_digraph.html#7`

Riferimenti bibliografici

LATEX3 PROJECT TEAM (2005). «LATEX 2ε font selection». `http://www.ctan.org/tex-archive/macros/latex/doc/fntguide.pdf`.

BECCARI, C. (2007). «Guida all'arte della composizione tipografica con LATEX». `http://www.guit.sssup.it/downloads/GuidaGuIT.pdf`.

KNUTH, D. E. (1984). *The TEXbook*. Addison-Wesley, Reading, MA, USA.

— (1990). «Virtual fonts: More fun for grand wizards». *TUGboat*, **11** (1), pp. 13–23.

LEHMANN, P. (2004). «The Font Installation Guide». `http://www.ctan.org/tex-archive/info/Type1fonts/fontinstallationguide/fontinstallationguide.pdf`.

MITTELBACH, F., GOOSSENS, M. *et al.* (2004). *The LATEX companion*. Addison Wesley, 2ª edizione.

ZANNARINI, E. e VAVASSORI, E. G. (2005). «LATEX e i font: installazione pratica». In *Secondo convegno nazionale su TEX, LATEX e tipografia digitale*. Pisa, pp. 107–122.

▷ Massimiliano Dominici
`mlgdominici@interfree.it`

Guidelines for Bibliographical Citations in LaTeX

Jean-Michel Hufflen

Abstract

After a short overview of the schemes used for bibliographical citations, we give some guidelines to use some packages of LaTeX 2ε and bibliography styles of BibTeX in order to write *adaptable* citations, i.e., texts where switching a citation scheme to another is easy.

Keywords LaTeX, BibTeX, ConTeXt, methodology, `bib` module, bibliography citation schemes, natbib, jurabib, camel, opcit, amsrefs, MlBibTeX.

Sommario

Dopo una breve panoramica degli schemi usati per le citazioni bibliografiche, l'articolo fornisce alcune linee guida per l'utilizzo dei principali pacchetti di LaTeX 2ε e stili bibliografici di BibTeX, allo scopo di ottenere citazioni *flessibili*, ovvero testi in cui risulti facile passare da uno schema di citazioni ad un altro.

Parole chiave BibTeX, ConTeXt, metodologia, modulo `bib`, schemi per citazioni bibliografiche, natbib, jurabib, camel, opcit, amsrefs, MlBibTeX.

1 Introduction

The TeX typesetting engine (Knuth, 1984) has succeeded in implementing a clear separation between layout and structure and the word processors derived from it—in particular, LaTeX (Lamport, 1994)—are unrivalled within this domain. However, if users wish to take as much advantage as possible of this feature, they should pay attention to write *adaptable* documents. For example, to put down a rule box between two paragraphs, as wide as possible, it is preferable to use:

```
\rule{\columnwidth}{...}
```

rather than a absolute length given in centimeters (or in another length unit) for the first argument. This allows such a rule to be adapted to the body's width, and the `\columnwidth` command is better than `\textwidth`, because it prevents the possible use of a two-column option (see (Mittelbach and Goossens, 2004, § 2.1) about this option), as in the style used in the present proceedings. Other examples, more related to semantic markup, can be found in (Hufflen, 2006b).

This article also explores methodology, but about bibliographical citations. It does not replace the good descriptions of bibliographical citation schemes and their implementation by means of LaTeX 2ε packages given in (Mittelbach and Goossens, 2004, Ch. 12) and in more specialised reference manuals. It just aims to study how to write bibliographical citations according to a *flexible* way, especially when a word processor is used in conjunction with a bibliography processor. A good example of such a cooperation is given by LaTeX and BibTeX (Patashnik, 1988a), the bibliography processor most commonly used with that word processor. This subject is not artificial: there are several citation schemes used throughout written documents. Often an author is familiar with a particular citation scheme when writing an article or a book, and may be asked by a publisher to rewrite this work by using another scheme. This *modus operandi* may be more difficult than planned.

First, we give a short overview of the citation schemes used, then we show how some bibliography styles of BibTeX and packages of LaTeX 2ε can put them into action. In a fourth section, we give some 'tricks' in order for bibliographical citations to be easily adaptable to most schemes. Then we explain how new bibliography styles should be built.

Reading this article requires a good practice of LaTeX and BibTeX, but as end-users do. It is not needed to know how to write a new bibliography style for BibTeX or new advanced commands for LaTeX. Let us only recall that when LaTeX and BibTeX are used in conjunction, bibliographical *entries* are given in bibliography database (.bib) files—an example is given in Figure 1—and can be referred by means of the `\cite` command of LaTeX. This command's argument must be a key used within such a .bib file, e.g. `\cite{robeson1974c}`. Then BibTeX searches these .bib files and builds a .bbl file containing bibliographical *references*[1] LaTeX can process. This .bbl file, containing the 'References' section usually put at a document's end, uses the `thebibliography` environment and the `\bibitem` command. See (Mittelbach and Goossens, 2004, § 12.1.2 & 12.1.3) for more details.

2 Citation schemes

Roughly speaking, bibliographical citations serve two purposes:

- they can refer to a work where more information could be found,

1. Here we use a precise terminology, which is defined within MlBibTeX ('**M**ulti**L**ingual BibTeX'), our reimplementation of BibTeX (2003): an entry (resp. reference) belongs to a .bib (resp. .bbl) file.

```
@BOOK{herbert-anderson2000,
        AUTHOR = {Brian Herbert and Kevin J.
                  Anderson},
        TITLE = {House {Atreides}},
        PUBLISHER = {Bantam},
        SERIES = {Prelude to Dune},
        NUMBER = 1,
        YEAR = 2000,
        MONTH = feb}

@BOOK{robeson1974b,
        AUTHOR = {Kenneth Robeson},
        TITLE = {The Crimson Serpent},
        PUBLISHER = {Bantam},
        SERIES = {Doc Savage Series},
        NUMBER = 78,
        YEAR = 1974,
        MONTH = oct}

@BOOK{robeson1974c,
        AUTHOR = {Kenneth Robeson},
        TITLE = {The Devil Genghis},
        PUBLISHER = {Bantam},
        SERIES = {Doc Savage Series},
        NUMBER = 79,
        YEAR = 1974,
        MONTH = nov}
```

FIGURE 1: Bibliography sample.

- or they can acknowledge anterior or comparable works.

In both cases, we require that references to citations are unambiguous, so looking for them within the 'References' section should be easy. Some examples could be:

> [3] is known as one of the best adventures of Doc Savage. [...] Brian Herbert and Kevin J. Anderson continued the *Dune* saga [1].

where a 'plain'[2] style is used, that is, references are labelled with numbers.

The style of the first example is controversial: some people think that a text should remain intelligible if references to bibliographical citations are removed. A good rewriting of this sentence would be:

> *The Devil Genghis* [3] is known as one of the best adventures of Doc Savage.

Let us remark that the title has been typed 'manually' in this previous sentence. In fact, we can think that there is information redundancy between the title both typed and given in the 'References' section. If there is a mistake about this title, it must be fixed in several places: the bibliography database file and all the occurrences within the text's body.

2. According to BIBTEX's terminology.

The second example does not meet this criticism, that sentence makes sense even if the citation is removed, but let us rewrite it using a 'long' style, that is, references are labelled with authors' names, followed by the publication's year:

> Brian Herbert and Kevin J. Anderson continued the *Dune* saga [Herbert & Anderson 2000].

First, the same problem of information redundancy occurs, about the authors' names. Second, the citation could omit them, since they are given earlier in the sentence:

> Brian Herbert and Kevin J. Anderson continued the *Dune* saga [2000].

Now let us come to a short synthesis of the main citation schemes used throughout printed documents. As mentioned in (Butcher, 1992, § 10), there are four schemes:

number-only publications are sequentially numbered in the 'References' section, and citations refer to these numbers;

author-date[3] citations use authors' names, followed by the publication's year; if an author published several works in a year, that year is suffixed with lowercase letters (e.g., '1974a', '1974b'); in addition, references to citations can be abridged: only the year is given if the author's name is known, and multiple citations concerning the same author supply this name only once (e.g., '[Robeson 1974a, 1974b]' rather than '[Robeson 1974a, Robeson 1974b]'); more details about this scheme can be found in (Chicago, Ch. 16);

author-number this system is similar to the author-date system, except that each author's publications are numbered and citations use authors' names, followed by this number, e.g., '[Robeson (2)]';

short-title references are given directly in the text at the first mention, the others may consist of an abridged form using the author's name only, followed by an abbreviation of the title if there are several works of this author[4]; a full bibliography at the document's end may sum up all the references, it can also be omitted, since a complete specification of each reference is provided either inside the text, or in footnotes[5].

3. Also known as 'Harvard system'.

4. From our personal point of view, 'short-title' is not good terminology for this system, because titles may be omitted for repeated references, if there is no ambiguity. In other words, the short-title system does not use titles always.

5. This last style is most used within this scheme and often called 'footnote-referencing'.

Herbert, Brian and Anderson, Kevin J. continued the *Dune* saga[1].

1. Herbert, Brian and Anderson, Kevin J. House Atreides. Bantam, February 2000, Prelude to Dune 1.

FIGURE 2: Citation as a footnote.

To go on with our previous examples, the first could be rewritten using the short-title system:

Robeson, *The Devil Genghis* is...

if this book has already been referred in a text where other books of Kenneth Robeson are cited. Let us remark that this sentence must be reformulated differently if references are given as footnotes. In this last case, our second example looks better, as shown in Figure 2.

3 Bibliography styles and packages

The LaTeX distribution provides a rich set of bibliography styles usable with BibTeX, summarized in (Mittelbach and Goossens, 2004, Table 13.4). Let us connect them to the schemes described in § 2. There is the 'plain' family, putting the number-only scheme, including styles such as plain, abbrv, and acm. We can consider that the 'alpha' family also puts this scheme into action. Labels are not numbers, but keys retaining some letters at the beginning of names, followed by the publication's year. For example, the alpha style would produce the keys '[HA00]', '[Rob74a]', '[Rob74b]' for the three entries of Figure 1. These keys are atomic, in the sense that it is impossible to split them into the information about author and year, so this style does not have the expressive power of an author-date system. The same for the styles belonging to the 'long' family[6] (see Hufflen et al., 1998, § 4.3.3).

Some bibliography styles of BibTeX were developed as early attempts of the author-date scheme: authordate{1–4}, chicago, harvard, as mentioned in (Mittelbach and Goossens, 2004, § 12.3.1). In addition to each bibliography style, a LaTeX 2ε package provided more functionalities in order to cite only the authors of a reference—using an abbreviated form or *in extenso*—or the date only, or both.

Now, the best implementation of this scheme is the natbib[7] package (see Mittelbach and Goossens, 2004, § 12.3.2). Let us consider again our second example:

```
\citeauthor{herbert-anderson2000}
continued the \emph{Dune} saga
\citeyearpar{herbert-anderson2000}.
```

6. These styles are old and no longer provided in modern distributions of LaTeX. With some stylistic variants, they behaved as described in § 2 (see our examples).

7. **NAT**ural sciences **BIB**liography.

```
\documentclass{article}

\usepackage{jurabib}
\jurabibsetup{super,authorformat=and}

\begin{document}

\citefield{author}{herbert-anderson2000}
continued the \emph{Dune} saga
\fullcite{herbert-anderson2000}.

\bibliography{...}
\bibliographystyle{jurabib}

\end{document}
```

FIGURE 3: How to get the result shown in Fig. 2.

the \citeyearpar command putting the year between delimiters, whereas \citeyear does not put them:

```
They've continued it since
\citeyear{herbert-anderson2000}.
```

We are close to the expected result, except that only last names are retained, first names are dropped out:

> Herbert and Anderson continued the *Dune* saga [2000]. They've continued it since 2000.

The short-title scheme has been fully implemented by the jurabib package (see Mittelbach and Goossens, 2004, § 12.5.1). This package is easily customisable by means of the \jurabibsetup command, as shown in Figure 3. Another implementation of the short-title scheme is given by the camel package (see Mittelbach and Goossens, 2004, § 12.5.2), but it uses an enlarged version of BibTeX. A package close to this scheme is opcit (Garcia, 2006), since it implements citations as footnotes, as often within this scheme. By default, this last package automatically uses its own bibliography style, opcit.bst.

4 Adaptable citations

4.1 Methodology

Let us recall that we aim to write adaptable citations in LaTeX source file, in order to prevent a change to another scheme if an article has already been written. On another point, we aim to avoid information redundancy and attempt to extract as much information as possible from the metadata given in bibliography database files, and transcribed in the result files generated by BibTeX. Here we are interested only superficially in the look of the citation keys and the 'References' sections: besides, many packages allow end-users to en-

```
\bibitem[Herbert and Anderson(2000)]{herbert-anderson2000}
Brian Herbert and Kevin~J. Anderson.
\newblock \emph{House {Atreides}}.
\newblock Number~1 in Prelude to Dune. Bantam, February 2000.

...

\bibitem[Robeson(1974{\natexlab{b}})]{robeson1974c}
Kenneth Robeson.
\newblock \emph{The Devil Genghis}.
\newblock Number~79 in Doc Savage Series. Bantam, November 1974{\natexlab{b}}.
```

FIGURE 4: References produced by the plainnat bibliography style.

hance them (see (Mittelbach and Goossens, 2004), throughout Chapter 12).

Our first advice is to avoid the sentences that need the full mention of a title, unless such style is absolutely necessary. As far as we know, only one package provides a command returning the title only, analogously to the commands `\citeauthor` and `\citeyear` of the natbib package. If we look into the jurabib package, its `\citetitle` command puts titles unconditionally, but after authors' names[8]). This package also provides a `\citetitleonly` command, but the title is given as a footnote if references are to be given in footnotes. In other words, this command would be unsuitable for our first example (concerning *The Devil Genghis*) within the source text given in Figure 3. The only way to get a title inside a text, regardless of the citation style—as footnotes or in-text—is '`\citefield{title}{...}`'—see another example of using this command in Figure 3—provided by jurabib.

In fact, we think that authors should use the short-title scheme only if they are asked for that. Of course, this scheme offers undisputable advantages and is widely used in humanities. Moreover, the two implementations in LATEX of this scheme are very powerful. But it is difficult to adapt texts written using this scheme if we have to switch it to another. It is even difficult to switch a package implementing this scheme to the other. First, they use specific bibliography styles for BIBTEX: e.g., jox, jurabib, jureco, jurunsrt for jurabib. Second, the camel engine uses its own commands for citations and bibliography management. In addition, jurabib and camel have been developed in order to process juridical works: they provide features unknown or rarely used in other fields. To sum up, the texts written with these packages may be very difficultly adaptable to other schemes, because of stylistic variants or because the commands used for citations are different.

For adaptable citations, the best solution is to use the natbib package as far as possible because it can produce author-date references as well as number-only ones. Using the numbers option:

```
\usepackage[numbers]{natbib}
```

causes bibliographical references to be labelled with numbers, as in the number-only scheme, whereas the commands `\citeauthor`, `\citeyear`, and `\citeyearpar` are still available. As shown in our previous examples, the `\citeyear` command should be used when the year is to be put down in any case. When only the year information is given because the author information is already know, a 'trick' is to define a new command:

```
\newcommand{\citeend}[1]{%
 \ifNAT@numbers\citep{#1}\else%
  \citeyearpar{#1}\fi}
```

(let us remark that the commands `\makeatletter` and `\makeatother` may be needed before and after this definition: see (Mittelbach and Goossens, 2004, § A.1.1) for more details). This `\citeend` command, used within a sentence such as:

```
... continued the \emph{Dune} saga
\citeend{herbert-anderson2000}.
```

will produce the citation of a numerical label if the numbers option has been selected, the year otherwise. In any cases, delimiters are put down.

For complete references, use the `\citep` command rather than `\cite` because the former always produces references between delimiters, as numbers if the numbers option is selected, as author-date information otherwise.

This *modus operandi* is also suitable for *unsorted* bibliographies, in which case the unsrtnat bibliography style should be used. Let us recall that in such styles, bibliographical items are not sorted with respect to the alphabetical order of authors or organisations, the order of such items in the 'References' section is the order of first citations of these items throughout the document. If references are labelled with numbers[9], finding an item within an unsorted bibliography is easy, but that becomes

8. See Footnote 4, p. 104.

9. In France, such a scheme is still used for Medicine

```
\bibitem[{Herbert\jbbtasep Anderson\jbdy {2000}}%
         {}%
         {{0}{}{book}{2000}{}{}{}{}%
          {Bantam\bibbdsep {} 2000}}%
         {{House {Atreides}}%
          {}{}{2}{}{}{}{}{}}%
        ]{herbert-anderson2000}
 \jbbibargs {\bibnf {Herbert} {Brian} {B.} {} {}\Bibbtasep \bibnf {Anderson}
  {Kevin~J.} {K.~J.} {} {}} {Brian HerbertKevin~J. Anderson} {aus} {\bibtfont
  {House {Atreides}}\bibatsep\  \apyformat {Bantam\bibbdsep {} \febname\ 2000}
  \numberandseries {1}{Prelude to Dune}} {\bibhowcited} \jbdoitem
  {{Herbert}{Brian}{B.}{}{}; {Anderson}{Kevin~J.}{K.~J.}{}{}} {} {}
  \bibAnnoteFile {herbert-anderson2000}

...

\bibitem[{Robeson\jbdy {1974}}%
         {}%
         {{1}{}{book}{1974\el {b}}{}{}{}{}%
          {Bantam\bibbdsep {} 1974\el {b}}}%
         {{The Devil Genghis}%
          {}{}{2}{}{}{}{}{}}%
        ]{robeson1974c}
 \jbbibargs {\bibnf {Robeson} {Kenneth} {K.} {} {}} {Kenneth Robeson} {au}
  {\bibtfont {The Devil Genghis}\bibatsep\  \apyformat {Bantam\bibbdsep {}
  \novname\ 1974\bibel {b}} \numberandseries {79}{Doc Savage Series}}
  {\bibhowcited} \jbdoitem {{Robeson}{Kenneth}{K.}{}{}} {} {} \bibAnnoteFile
  {robeson1974c}
```

FIGURE 5: References produced by the jurabib bibliography style.

difficult if a large bibliography with unordered labels is to be searched. In addition, checking if a particular work is cited within a 'References' section may be tedious.

4.2 Towards more powerful bibliography styles

The citation scheme used by LaTeX defaults to number-only, in which case the `\bibitem` command only has a mandatory argument:

```
\bibitem{robeson1974c}...
```

If we use a style belonging to the 'alpha' or 'long' family, an optional argument—built by BibTeX—specifies which label is to be put in front of the corresponding reference:

```
\bibitem[Rob74a]{robeson1974c}...
```

When a bibliography style suitable for the natbib package is used, this optional argument is used by the functions of the natbib package to retrieve the different parts of the author-date information: we show how this optional argument is built in Figure 4.

The structure passed from BibTeX to LaTeX is much more expressive when we consider bibliography styles suitable for the jurabib package. Figure 5 shows how the jurabib bibliography style processes the entries of Figure 1, this .bbl file being usable by the source text given in Figure 3. This kind of structure would be very tedious to produce manually. But the jurabib package allows much customisation and such complexity is the price to pay. For example, '`\jbbtasep`' stands for '**B**etween **T**wo **A**uthors **S**eparation', '`\jbdy`' for '**D**ummy **Y**ear'[10], the '`book`' information is needed since jurabib allows a kind of entries to be processed specifically, ...

However, transmitting a structured text to the packages dealing with bibliography citations is the only way towards more powerful commands for citations. In fact, a modern version of the `\bibitem` command should not only provide a label, but also encompass all the metadata for a bibliographic item. The first implementation of this idea is given by the amsxport bibliography style (Downes, 2000), which generates output files like the example of Figure 6. Let us remark that this style's main purpose is not the definition of commands accessing

Ph.D.s, but—strangely—not for the documents written by nurses, where the short-title scheme is usual. As far as we know, sorted bibliographies are commonly used within other scientific fields.

10. By default, the `\jbdy` command skips the next token, and `\jbbtasep` inserts a '/' character. This last command can be redefined by means of the `\jurabibetup` command, as shown in Figure 3.

```
\begin{bibdiv}
\begin{biblist}

\bib{herbert-anderson2000}{book}{
      author={Herbert, Brian},
      author={Anderson, Kevin~J.},
       title={House {Atreides}},
      series={Prelude to Dune},
   publisher={Bantam},
        date={2000},
      number={1},
}

...

\end{biblist}
\end{bibdiv}
```

FIGURE 6: References produced by the amxport bibliography style.

```
\startpublication[k=herbert-anderson2000,
t=book,
a={{Herbert},{Anderson}},y=2000,
n=1,s=HA00]
\author[]{Brian}[B.]{}{Herbert}
\author[]{Kevin~J.}[K.~J.]{}{Anderson}
\pubyear{2000}
\title{House {Atreides}}
\series{Prelude to Dune}
\volume{1}
\pubname{Bantam}
\month{2}
\stoppublication

...

\startpublication[k=robeson1974c,t=book,
a={{Robeson}},y=1974b,
n=3,s=Rob74b]
\author[]{Kenneth}[K.]{}{Robeson}
\pubyear{1974\maybeyear{b}}
\title{The Devil Genghis}
\series{Doc Savage Series}
\volume{79}
\pubname{Bantam}
\month{11}
\stoppublication
```

FIGURE 7: References used by the bib module of ConTEXt.

particular parts of a reference within a text's body. It aims to allow the specification of formatting bibliographical references using LATEX-like syntax. That is done by means of the functionalities of the amsrefs package.

A more modern implementation of this *modus operandi* is the bib module of ConTEXt (Hoekwater, 2006), the result of processing our bibliography sample being given in Figure 7. In ConTEXt (see Hagen, 2001, § 2.2), commands like '`\start...`' and '`\stop...`' are analogous to environment delimiters in LATEX ('`\begin{...}`' and '`\end{...}`'). The structured information put between the commands `\startpublication` and `\stoppublication` may be customised as described in (Hoekwater, 2006, § 2.2) by means of the `\setuppublicationlist` command. In particular, this command allows authors' names to be typeset *in extenso* or abbreviated, but such customisation exists for the 'References' section, not for citations of authors' names inside a text. The bib module[11] also provides an advanced command for citations (see Hoekwater, 2006, § 3), e.g.:

```
\cite[author][robeson1974b]
```

Let us remark that only some options are available: author, authoryear, year, ... but this command cannot put only a title. The layout of the texts produced by this command may be customised by means of the `\setupcite` command. In Figure 8, you can see how to get a publication's year with delimiters: by associating characters to the keywords left and right within an accurate `\setupcite` command. To get the same information without delimiters later in the text, use this command again with these two keywords associated to blank values.

11. Roughly speaking, a ConTEXt module is analogous to a LATEX 2_ε package.

Roughly speaking, such an exercise may be viewed as worthwhile... or tedious. In fact, we personally consider that the approach of this module is promising, but it could be improved. Last, it is tightly related to ConTEXt, which does not use the same commands than LATEX.

Another point is related to multilinguism: if a citation command does not put keys, but text fragments, they should be hyphenated correctly if need be. By 'correctly', we mean 'using the right patterns', depending on languages. This is planned by the functions of the jurabib package, although all the information concerning a reference is supposed to be expressed in the same language. For example, an Italian author may write a text in English, in which case the title would be properly hyphenated, but not the author's name. As another example related to multilinguism, the connection between successive co-authors' names is also language-dependent: 'and' in English, '*e*' in Italian, ... The jurabib package and the bib module can deal with such language-dependent features, the natbib package could be improved this way.

5 Conclusion

We personally used the number-only scheme for a long time. Then we had to rewrite several of our texts using the author-date scheme. We also helped some people to prepare written documents

```
\usemodule[bib]

\setupbibtex[database=...]

\starttext

\setupcite[author][left=,right=]
\setupcite[year][left=(,right=)]
\cite[author][herbert-anderson2000] have
continued the {\em Dune} saga
\cite[year][herbert-anderson2000].
\setupcite[year][left=,right=]
They've continued it since
\cite[year][herbert-anderson2000].

\placepublications

\stoptext
```

FIGURE 8: Example using ConTEXt's bib module.

about other fields than those we have usually dealt with. Some adaptations were difficult, especially about unsorted bibliography styles and texts using the short-title scheme. Now we are aware of the influence of citation schemes on the style of writing and began to experiment some solutions.

On another point, we are presently rewriting the bibliography styles usable with the packages natbib and jurabib, taking as much advantage as possible of the new functionalities provided by MlBibTeX, our reimplementation of BibTeX (see App. A). This work has definitely convinced us that the present form of the `\bibitem` command was no longer suitable for modern requirements. As a possible solution, we plan to investigate the use of LuaTeX (Hagen, 2006), a new typesetting engine built out of TeX, mixing the expressive power of TeX's features and service provided by a modern programming language. Within this framework, a bibliography processor could result in structures suitable for the Lua programming language (Ierusalimschy, 2006) when entries are processed. Then, when the word processor deals with a text, bibliographical references could be formatted by running fragments written in Lua.

A Using MlBibTeX

MlBibTeX is a reimplementation of BibTeX with particular focus on multilingual features (Hufflen, 2003). As we explain in (2006c), it uses XML[12] as a central formalism and bibliography styles may be written using a variant of XSLT[13]. A compatibility mode allows bibliography styles of BibTeX to be run with MlBibTeX (2006a), although some change would be needed to run the bibliography styles associated with the jurabib package[14].

Concerning the problems raised by the different citation schemes, MlBibTeX is able to provide the following solutions.

- It eases the specification of styles according to the author-number scheme[15]. These styles could be used with the natbib package[16].
- More generally, we think that the new language for bibliography styles eases the programming of labels for references. For example, we can associate the two citation keys '[Rob 74a]' and '[Rob 74b]' with the two entries of Figure 1 whose author is Kenneth Robeson, according to the alpha bibliography style. But a variant of this style could associate the two labels '[Rob 74]' and '[Rob 74a]' with these two items, that is, an additional letter is used only from the second work if an author published several documents in a year. Such variants are easy to program with MlBibTeX, in comparison to what should be done with the language of BibTeX's bibliography styles.
- MlBibTeX provides additional markup when some information—person names and titles especially—belongs to a language different from the global language of an entry. An example is given in Figure 9: this bibliographical entry concerns a work written in Italian, so the author's name and the title are supposed to be in Italian. This work is included in an international anthology whose two editors are American and French. In addition, the anthology's title is given in French. These informations about languages are given in Figure 9, this entry being suitable for MlBibTeX. However, some rewriting of the natbib package would be needed in order for this package to take advantage of this feature.

Acknowledgements

I have learned very much whenever I helped some people to play with BibTeX and the citation commands of LaTeX. Initially, some of these people did not plan to use these two programs and had already begun to write important parts of their

12. e**X**tensible **M**arkup **L**anguage. Readers interested in an introduction to it can refer to (Ray, 2001).

13. e**X**tensible **L**anguage **S**tyle **T**ransformations. This language is used to write transformations of XML texts (W3C, 1999).

14. That is because MlBibTeX and the styles suitable for jurabib both deal with a `LANGUAGE` field inside bibliographical entries, but this field obeys different conventions.

15. Although developing such styles using the language of bibliography styles of BibTeX (Patashnik, 1988b) should be possible. However, that has not been done yet, as mentioned in (Mittelbach and Goossens, 2004, § 12.4).

16. There is no current support of this scheme for LaTeX, unless we accept that references are numbered globally, in which case the natbib package can be used with the numbers option (see Mittelbach and Goossens, 2004, § 12.4.1).

```
@INPROCEEDINGS{evangelisti2000,
        AUTHOR = {Valerio Evangelisti},
        TITLE = {Paradice},
        EDITOR = {[Robert Silverberg] :
                  english and
                  [Jacques Chambon] : french},
        BOOKTITLE = {[Destination 3001] :
                     french},
        ...
        LANGUAGE = italian}
```

FIGURE 9: How to specify 'foreign' information in MlBIBTEX.

works. Thanks to them, because such cases gave me the idea of this paper. I am also grateful to Massimiliano Dominici, who has written the Italian translation of the abstract. Last, thanks to the anonymous referee, who suggested me significant improvement to the first version.

References

Judith Butcher. *Copy-Editing. The Cambridge Handbook for Editors, Authors, Publishers.* Cambridge University Press, 3rd edition, 1992.

Chicago. The Chicago manual of style. The University of Chicago Press, 1993. The 14th edition of a manual of style revised and expanded.

Michael Downes. The amsrefs LATEX package and the amsxport BIBTEX style. *TUGboat*, 21(3): 201–209, September 2000.

Federico Garcia. *opcit, a Package for Footnote-Style Bibliographical References.* ftp://ctan.tug.org/tex-archive/macros/latex/contrib/opcit/opcit.pdf, September 2006.

Hans Hagen. *ConTEXt, the Manual.* http://www.pragma-ade.com/general/manuals/cont-enp.pdf, November 2001.

Hans Hagen. LuaTEX: Howling to the moon. *Biuletyn Polskiej Grupy Użytkowników Systemu TEX*, 23:63–68, April 2006.

Taco Hoekwater. *ConTEXt. Module Documentation. Bibliographies.* http://dl.contextgarden.net/modules/t-bib/doc/context/bib/bibmod-doc.pdf, March 2006.

Jean-Michel Hufflen. MlBIBTEX's version 1.3. TUG*boat*, 24(2):249–262, July 2003.

Jean-Michel Hufflen. BIBTEX, MlBIBTEX and bibliography styles. *Biuletyn* GUST, 23:76–80, April 2006a. In *BachoTEX 2006 conference.*

Jean-Michel Hufflen. Writing structured and semantics-oriented documents: TEX vs XML. *Biuletyn* GUST, 23:104–108, April 2006b. In *BachoTEX 2006 conference.*

Jean-Michel Hufflen. MlBIBTEX architecture. *ArsTEXnica*, 2:54–59, October 2006c. In GUIT 2006 meeting.

Jean-Michel Hufflen, Denis B. Rœgel, and Karl Tombre. Guide local (LA)TEX du LORIA. Millésime 1998. Technical Report 98–R–214, LORIA, September 1998.

Roberto Ierusalimschy. *Programming in Lua.* Lua.org, 2 edition, March 2006.

Donald Ervin Knuth. *Computers & Typesetting. Vol. A: The TEXbook.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.

Leslie Lamport. *LATEX: A Document Preparation System. User's Guide and Reference Manual.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1994.

Frank Mittelbach and Michel Goossens. *The LATEX Companion.* Addison-Wesley Publishing Company, Reading, Massachusetts, 2 edition, August 2004.

Oren Patashnik. *BIBTEXing.* Part of the BIBTEX distribution, February 1988a.

Oren Patashnik. *Designing BIBTEX Styles.* Part of the BIBTEX distribution, February 1988b.

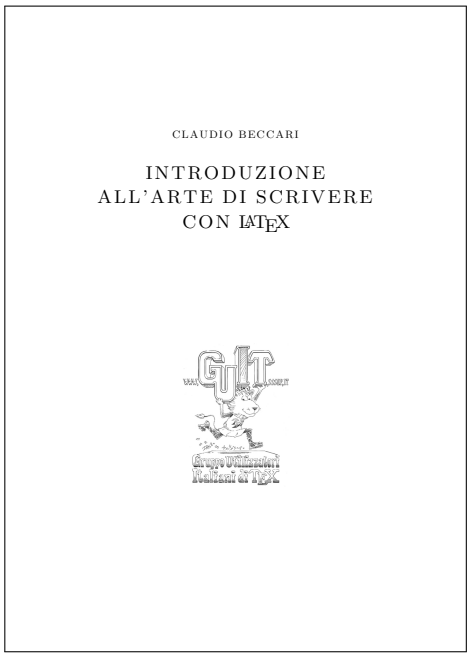
Erik T. Ray. *Learning* XML. O'Reilly & Associates, Inc., January 2001.

W3C. XSL *Transformations (*XSLT*). Version 1.0.* http://www.w3.org/TR/1999/REC-xslt-19991116, November 1999. W3C Recommendation. Edited by James Clark.

▷ Jean-Michel Hufflen
LIFC (EA CNRS 4157)
University of Franche-Comté
16, route de Gray
25030 BESANÇON CEDEX
FRANCE
hufflen@lifc.univ-fcomte.fr

Eventi e novità

Presentazione del libro di Claudio Beccari *Introduzione all'arte di scrivere con LaTeX*



Il Gruppo Utilizzatori Italiani di TeX è lieto di presentare la nuova introduzione all'uso di LaTeX, realizzata dal Prof. Claudio Beccari. Come sottolinea l'autore nella prefazione, non si tratta di un manuale nel senso classico del termine. Piuttosto, lo scopo dell'opera è quello di fornire un approccio alla scrittura con LaTeX basato sui principi della composizione professionale, incoraggiando il lettore a pensare in termini di buon disegno tipografico, invece di inseguire il singolo dettaglio grafico.

Il libro offre una panoramica completa e approfondita di tutti gli aspetti della composizione con LaTeX, riuscendo nell'intento di rendere chiari al lettore anche gli argomenti più complessi, e tenendo sempre un occhio puntato sul rispetto delle (buone) norme tipografiche.

Il testo è corredato di alcune utili appendici, tra cui un riepilogo della sintassi di LaTeX che può a buon diritto essere considerato una sorta di "indice concettuale" dell'intero volume.

Il testo, di cui alcune copie sono state rese disponibili durante il GuIt *meeting*, è liberamente scaricabile all'indirizzo `http://www.guit.sssup.it/downloads/`.

Asian TUG meeting

Il 25 e 26 Gennaio 2008 si svolgerà presso la Kongju National University (KNU), nella Corea del Sud, la Asian TeX Conference 2008 (AsiaTeX 08).

AsiaTeX 08 è organizzato dalla Korean TeX Society (KTS) e dal dipartimento di Economia (KNU), e ospitato dalla Facoltà di Lettere e Scienze Sociali, KNU.

Per ulteriori informazioni: `http://conf.ktug.kr/2008/index.html`

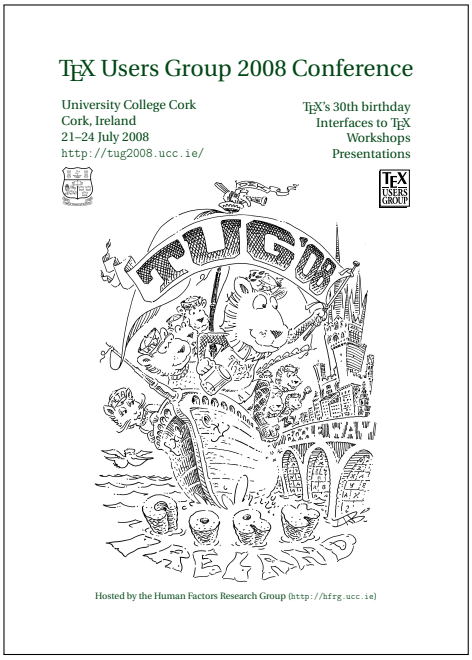
DANTE 2008

Il convegno DANTE 2008 si terrà dal 5 al 7 Marzo 2008, presso la Friederich-Schiller-Universtät di Jena.

Organizzano il convegno DANTE, e l'Isituto di Studi di Filologia e Linguistica Germanica della Friederich-Schiller-Universtät di Jena.

Per ulteriori informazioni: `http://www.dante.de/dante2008/`

TUG Conference 2008



Il convegno annuale 2008 del TeX Users Group si dal 21 al 24 luglio 2008 presso lo University College Cork, in Irlanda.

Nel 2008 cade il trentesimo anniversario della nascita di TeX. Temi del convegno saranno: Interfacce, LaTeX, ConTeXt, METAFONT, METAPOST e XeTeX.

Il convegno è ospitato dallo *Human Factors Research Group* presso il Dipartimento di Psicologia Applicata.

Per ulteriori informazioni: `http://www.ucc.ie/en/tug2008/`

Questa rivista è stata stampata
su carta ecosostenibile Repro R Ciclo
prodotta dalle cartiere Burgo.

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 2008. 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 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 `http://www.guit.sssup.it/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 4, Ottobre 2007