Nadia Garbellini

LaTeX facile

Guida all'uso

2010

Nadia Garbellini

LaTeX facile

Guida all'uso

seconda edizione riveduta e corretta

2010

PRESENTAZIONE

L'amica e brava Nadia Garbellini, autrice di questa bella e semplice guida per LaTeX, ha voluto aiutare quanti vogliono avvicinarsi a questo fantastico strumento, in grado di creare documenti di grande qualità e senza limitazioni grafiche di sorta. L'idea nasce quasi per caso, mentre si discuteva del manuale *Ubuntu facile* che stavo realizzando. In un primo momento, Nadia, pensava di realizzare un capitolo, del detto manuale su Ubuntu, dedicato a LaTeX. Però, andando avanti con la realizzazione dell'opera, mi resi conto, leggendo le varie revisioni, che sarebbe stato restrittivo relegare al rango di capitolo, una così completa e ed autonoma guida. Ma proprio perché in origine doveva essere un capitolo del manuale su Ubuntu, pregai Nadia di utilizzare una scrittura facile, immediata ed alla portata di tutti, anche di coloro che sono alle prime armi, proprio per conformarsi a criteri di semplicità del manuale. In definitiva abbiamo ritenuto di dare identità propria a questa bella e facile guida su LaTeX.

LaTeX è un linguaggio utilizzato per la composizione di testi, soprattutto scientifici ma anche letterari, che permette di ottenere risultati professionali: impaginazione, tabelle, note, bibliografa, e tutto quello che comunemente trovate in qualsiasi libro di testo può essere realizzato con il vostro computer... e un po' di voglia di imparare qualcosa di nuovo. Il secondo requisito è fondamentale: LaTeX, infatti, è molto diverso da ciò che siete abituati ad usare, probabilmente MicroSoft Word o OpenOffice Writer, sia nell'utilizzo che nella filosofia.

Contrariamente ai *word-processor*, che si basano sul paradigma WYSIWYG (What You See Is What You Get, "ciò che vedi è ciò che ottieni", ossia quello che vedete sul monitor è esattamente quello che ottenete stampando la pagina), LaTeX è un *text processor* e si lavora in modo WYSIWYM (What You See Is What You Mean, ossia "ciò che vedi è ciò che vuoi dire"). State utilizzando una sorta di linguaggio di programmazione, quindi ciò che vedete sul monitor è il 'codice' che immettete, grazie al quale potete concentravi sul quello che scrivete lasciando che LaTeX controlli tutti gli aspetti tipografici del testo che state componendo.

Eccovi alcuni validi motivi per usare a LaTeX.

1. LaTeX è un software completamente gratuito. Se siete degli utenti di Linux, questo dovrebbe essere già un buon motivo, in quanto si inserisce, perfettamente, nella filosofia di questo sistema operativo. Non solo: LATEX è stato sviluppato, in origine, proprio per Unix, che è il 'papà' di Linux (e di Mac OS X).

2. LaTeX è multipiattaforma, cioè funziona benissimo con qualunque sistema operativo stiate usando. Questo significa anche che i documenti scritti su Linux potranno essere tranquillamente letti su Windows.

3. I documenti scritti con il mark-up di LaTeX possono venire composti direttamente in pdf, oltre che in altri formati.

4. LaTeX è un ottimo strumento non solo per scrivere, ma anche per disegnare, fare tabelle e grafici, fare presentazioni, eccetera.

5. LaTeX vi permetterà di imparare a ragionare con la stessa logica che serve a costruire un programma informatico.

Addì, 18 settembre 2008 Prof. ing. Antonio Cantaro

Indice

Capitolo 1

Che cos'è LaTeX?

1.1 Introduzione

LaTeX è un linguaggio utilizzato per la composizione di testi, soprattutto scientifici ma anche letterari, che permette di ottenere risultati professionali: impaginazione, tabelle, note, bibliografia, e tutto quello che comunemente trovate in qualsiasi libro di testo può essere realizzato con il vostro computer... e un po' di voglia di imparare qualcosa di nuovo.

Il secondo requisito è fondamentale: LaTeX infatti è molto diverso da ciò che siete abituati ad usare – probabilmente MicroSoft Word o OpenOffice Writer – sia nell'utilizzo che nella filosofia.

1.1.1 Un po' di storia

LaTeX è basato su un motore di tipocomposizione chiamato TeX. I due programmi sono stati sviluppati inizialmente in parallelo, sebbene il primo sia basato sul secondo.

Donald Ervin Knuth, docente matematica e di informatica all'Università di Stanford, cominciò a lavorare a TeX nel 1977, principalmente allo scopo di disporre di uno strumento per la composizione di formule matematiche visto che i suoi libri venivano composti, volume dopo volume, sempre peggio a causa della caduta di professionalità dei compositori presso le case editrici di testi scientifici. All'epoca, inoltre, gli autori dovevano scrivere tutto il 'manoscritto' con la macchina da scrivere, in maniera estremamente laboriosa, e sia i grafici sia molti simboli dovevano essere aggiunti a mano alla fine.

Il progetto di LaTeX invece è stato iniziato, alla fine degli anni '70, da Leslie Lamport, che nel frattempo collaborava con Knuth. LaTeX in realtà non è un programma, ma un metodo di mark-up realizzato attraverso delle macro dichiarative e/o descrittive, che vengono poi interpretate dal motore di tipocomposizione che era, e in un certo senso rimane, il programma **tex**.

Ecco perché i due progetti hanno seguito per un po' di tempo strade parallele. Nel corso del tempo, dagli anni '80 ad oggi, entrambi i programmi sono stati costantemente migliorati e arricchiti di nuove funzionalità, fino a diventare quello che sono oggi. Naturalmente, i progetti sono costantemente in progresso, e molti sviluppatori concorrono ogni giorno alla creazione di nuovi strumenti.

1.1.2 L'etimologia

La parola TeX deriva da *τέχνη*, parola greca dal duplice significato di 'arte' e 'tecnica'. La 'X' di TeX e di LaTeX infatti, non si pronuncia alla latina, ma alla greca. Si tratta di un suono che non esiste nella nostra lingua (tranne che nella parlata fiorentina), ma che si pronuncia come il 'ch' in tedesco o la 'j' in spagnolo.

Conoscere l'etimologia della parola dovrebbe aiutarvi a capire qual è lo scopo di LaTeX: fornire uno strumento per l'arte della scrittura, che possa essere utilizzato anche per comporre testi scientifici.

Come abbiamo detto, LaTeX è nato soprattutto per scritti scientifici, ma nel corso del tempo sono stati introdotti pacchetti adatti a scrivere poesie, sceneggiature, versi in greco e latino (e moltissime altre lingue, compreso il cinese!) e persino partiture musicali.

1.2 Perché LaTeX è diverso?

La differenza principale, come abbiamo appena detto, risiede nella filosofia del programma: contrariamente ad altri word processor, che si basano sul paradigma WYSIWYG (What you See Is What You Get, 'ciò che vedi è ciò che ottieni': quello che vedete sul monitor è esattamente quello che ottenete stampando la pagina), con LaTeX si lavora in modo WYSIWYM – What you See Is What You Mean, 'ciò che vedi è ciò che vuoi dire'. State utilizzando una sorta di linguaggio di programmazione, quindi ciò che vedete sul monitor è il 'codice' che immettete, grazie al quale potete concentrarvi sul testo lasciando a LaTeX il compito di controllare tutti gli aspetti tipografici del testo che state componendo.

Probabilmente si tratta di un concetto un po' difficile da capire all'inizio. Andando avanti a leggere questa guida tutto dovrebbe diventare più chiaro, ma comunque è meglio fare subito un esempio.

Supponete di voler scrivere questa frase:

Non è *poi* **così** difficile

Per ottenere questo risultato, il codice è:

```
\textcolor{red}{Non} \textcolor{green}{è}
\emph{poi}\textbf{così} {\huge difficile}
```

Non spaventatevi!! Il carattere \ è semplicemente ciò che va messo davanti ad ogni comando per fare in modo che LaTeX lo consideri come tale. `\textcolor{`⟨*colore*⟩`}{`⟨*testo*⟩`}`, come avrete capito, è il comando che stabilisce il colore con cui scrivere determinate parole. Il colore va messo fra le prime parentesi, le parole da colorare fra le seconde. Questo è qualcosa che funziona con quasi tutti i comandi: si riferiscono alla parte di testo che è racchiusa tra le parentesi graffe che li seguono. Ma ne parleremo più diffusamente in seguito. `\emph{`⟨*parole*⟩`}` serve a evidenziare ⟨*parole*⟩ in corsivo (emph=emphasized, evidenziato); `\textbf{`⟨*parole*⟩`}` per scriverle in grassetto (bf=bold face) e `\huge␣`⟨*parole*⟩ per scriverle in corpo molto grande (huge=enorme); il segno ␣ serve per rendere visibile uno spazio.

1.3 Non è poi così difficile...

... anche se forse, a prima vista, può sembrarlo.

Le potenzialità di questo software sono enormi, e sfruttarle tutte richiede sicuramente una conoscenza molto, molto approfondita del suo linguaggio. Tuttavia, tale conoscenza si acquisisce piuttosto facilmente con la pratica, e per iniziare, vi assicuro, non occorre sapere molte cose. Inoltre, quasi tutti gli editor di testo[1] esistenti hanno dei menù a tendina che permettono di selezionare tutte, o quasi, le opzioni di base senza dover per forza conoscere il codice. Ma dopo qualche ora di utilizzo viene naturale inserirlo a mano!

Lo sforzo iniziale necessario a imparare ad usare questo software verrà presto ripagato dalle soddisfazioni che vi darà! Inoltre, il linguaggio di LaTeX è di fatto molto simile all'HTML. Questo significa che:

- per quanti di voi già usano un po' l'HTML, imparare il LaTeX sarà una passeggiata!
- per quanti di voi vorrebbero imparare l'HTML, LaTeX sarà un'ottima palestra.

1.4 Se non siete ancora convinti...

Se non siete ancora convinti che usare LaTeX sia una buona idea, proverò a fornirvi una serie di ragioni per cui, invece, dovreste esserlo, e assolutamente!

1. LaTeX è un software completamente gratuito. Se siete degli utenti di Linux, questo dovrebbe essere già un buon motivo, in quanto si inserisce perfettamente nella filosofia di questo sistema operativo. Non solo: LaTeX è stato sviluppato, in origine, proprio per Unix, che è il 'papà' di Linux (e di Mac OS X).
2. LaTeX è multipiattaforma, cioè funziona benissimo con qualunque sistema operativo stiate usando. Questo significa anche che i documenti scritti su Linux potranno essere tranquillamente letti su Windows.
3. I documenti scritti in LaTeX si possono velocissimamente convertire in formato pdf; o, meglio ancora, possono essere prodotti direttamente in pdf.
4. LaTeX è un ottimo strumento non solo per scrivere, ma anche per disegnare, fare tabelle e grafici, fare presentazioni, eccetera.
5. LaTeX vi permetterà di imparare a ragionare con la stessa logica che serve a costruire un programma informatico.

[1] Gli editor di testo sono semplicemente dei programmi, come il Blocco note per intenderci, che ci permettono di utilizzare LaTeX, cioè all'interno dei quali si scrive il codice. Ma anche di questo avremo modo di parlare in seguito...

Capitolo 2

Come cominciare

Come abbiamo già avuto modo di accennare alla fine del capitolo 1, per cominciare ad utilizzare LaTeX, oltre al software stesso, serve un editor di testo, che è l'interfaccia tramite la quale possiamo comporre i nostri testi.

Abbiamo anche detto che LaTeX è multipiattaforma. Questo però non significa che la stessa versione del programma, o lo stesso editor, funzioni con ogni sistema operativo.

Vediamo quindi nel dettaglio che cosa è necessario fare per avere sul nostro PC tutto ciò che ci occorre per metterci all'opera.

2.1 Procurarsi LaTeX

La cosa più semplice per ogni sistema operativo è quella di procurarsi il DVD TeXlive 2009 (o successivo) dalla associazione internazionale degli utenti di TeX (TUG); vedete il sito www.tug.org. Oppure scaricatevi l'immagine ISO da quel sito, masterizzatela, e poi usate il DVD così ottenuto come usereste quello comperato (a costo nominale) da TUG. Ricordate comunque che il sito di TUG è la sorgente principale di qualunque cosa sia collegato con il sistema TeX; essa attraverso un sistema di archivi collegati e sincronizzati fra di loro (e con i loro mirror) provvede a rendere disponibile gratuitamente tutto il materiale che può servire.

2.1.1 Per tutte le piattaforme

Se usate il DVD la prima cosa da scaricare è il programma perl **tlmgr** che vi permette di curare la manutenzione del vostro sistema TeX, di aggiornarlo, di arricchirlo di altri pacchetti, eccetera

Volendo, potete fare a meno del DVD e potete scaricare dal sito di TUG solo il programma perl **tlmgr** e usare questo programma anche per l'installazione iniziale. L'importante è poi disporre di in collegamento internet veloce, perché bisogna scaricare molti megabyte di materiale.

Questo, lo si sottolinea, è un procedimento che vale per tutte le piattaforme, anche se, volendo, ogni piattaforma ha suoi metodi particolari e installazioni particolari, tutte sempre completamente o parzialmente compatibili con la distribuzione ufficiale di TUG.

2.1.2 Linux

Se lavoriamo in ambiente Linux (voi avete probabilmente già cominciato ad usare, sicuramente con successo, Ubuntu 9.10), le cose sono estremamente semplici. Le istruzioni che seguono si applicano proprio ad Ubuntu, la distro che voi conoscete meglio.

La procedura è davvero semplice. Aprite Synaptic; cercate, e installate, texlive, texlive-base e texlive-common. Se avete qualche mega di spazio libero, potete installare anche texlive-full. La differenza tra la versione base e quella full è che la seconda contiene già tutti i pacchetti presenti per LaTeX.

La prima invece contiene solo i pacchetti, appunto, di base. Tutti gli altri pacchetti che potrebbero servirvi, andranno quindi installati di volta in volta, con una procedure molto semplice, che vedremo a tempo debito.

2.1.3 Mac OS X

Tutto quello che vi occorre è liberamente scaricabile all'indirizzo `http://www.tug.org/mactex/2009/downloading.html`.

La versione ufficiale di ogni anno cambia il valore '2009' nel valore dell'anno che vi interessa, anche se in verità l'aggiornamento avviene verso la fine dell'anno, per poter raccogliere tutti gli aggiornamenti che sono stati prodotti nell'anno in corso.

Come sempre, l'installazione su un Mac è semplicissima e completa, perché basta decomprimere il file *.zip* appena scaricato, leggere le semplici istruzioni, e cliccare sull'icona del 'pacchetto imballato'; tutto il sistema TeX viene installato completamente ed è immediatamente usabile, perché contiene anche i 'text editor' **TeXShop** e **TeXworks**, entrambi molto validi, anche se il secondo ha pochi anni di vita e quindi non ha ancora (fine 2009) la funzionalità completa.

L'uso del DVD con Mac diventa interessante se *non* volete eseguire una installazione completa, che per altro occupa molti megabyte di spazio sul disco fisso.

2.1.4 Windows

Gli utenti dei sistemi Windows generalmente non usano (a torto) il DVD o l'installazione attraverso il programma perl **tlmgr** ma si affidano a due altri metodi per installare la distribuzione MiKTeX.

Infatti la distribuzione di LaTeX più diffusa per Windows è senz'altro MiKTeX, che potete liberamente scaricare direttamente dal sito di Miktex all'indirizzo `http://miktex.org/`.

Alternativamente, se disponete del DVD o dell'immagine ISO masterizzata, potete usare l'installatore pdf ProTeX, che vi guida nelle varie piccole operazioni che il sistema MiKTeX richiede.

In entrambi i casi avete due opzioni: installare la versione 'di base' o quella completa.

2.2 Procurarsi un buon editor di testo

Esistono moltissimi editor di testo adatti a lavorare con LaTeX, alcuni specifici per LaTeX stesso, altri che possono essere utilizzati anche per altri linguaggi di

programmazione. In quanto segue, vi consiglierò un editor per ciascun sistema operativo, limitandomi a quelli liberamente scaricabili dalla rete. Installando il sistema TEX dal DVD o dall'immagine ISO masterizzata, tutte le piattaforme dispongono dell'editor **TeXworks**, ma ci sono altri ottimi editor descritti qui di seguito.

Generalmente i buoni editor adatti a LATEX lavorano con due finestre contemporaneamente sullo schermo; una contiene il testo LATEX da comporre, l'altra contiene il testo composto.

Va segnalato che una qualità importante per gli editor adatti a LATEX è quella di poter eseguire la ricerca diretta e inversa del testo: la ricerca diretta consiste nel ricercare nella finestra del testo composto il punto esatto corrispondente alla posizione del cursore nella finestra di composizione; la ricerca inversa serve per l'operazione opposta.

Quasi tutti i buoni editor consentono di spostarsi fra la finestra del testo da comporre e la finestra del testo composto quando questa è nel formato dvi, quello di default del sistema TEX; oggi (inizio 2010) solo gli editor **TeXShop** (specifico per Mac) e **TeXworks** (multipiattaforma) consentono di eseguire nativamente questi due tipi di ricerca quando la finestra del testo composto è in formato pdf.

2.2.1 Ubuntu

Aprendo 'Aggiungi/Rimuovi Applicazioni' dal menù, potete trovare moltissimi editor adatti a lavorare con LATEX. Alcuni però sono piuttosto difficili da usare e richiedono molta esperienza. Io vi consiglio invece di installare **Texmaker** (quello che sto usando io in questo momento; esso è multipiattaforma, nel senso che esistono anche le versioni per Windows e Mac) che è davvero semplice ed intuitivo da usare. Esso può essere scaricato da `http://www.xm1math.net/texmaker/texmaker_1.9.2-1_i386.deb`.

TeXworks per Ubuntu consente, come detto sopra, di eseguire la ricerca diretta e inversa anche quando il testo è composto direttamente in formato pdf.

2.2.2 Mac

Il pacchetto che scaricate dal link fornito nel paragrafo 2.1.3 è già comprensivo di un buon editor di testo. Quindi, se avete un Mac, avete già tutto quello che vi occorre.

2.2.3 Windows

Pur essendo disponibile l'editor **TeXworks**, in ambiente Windows l'editor più usato è probabilmente **WinEdt**. Tuttavia non si tratta di un software freeware, quindi il mio consiglio è di utilizzare **TeXmaker**, che è ottimo, e può essere scaricato dal sito `http://www.xm1math.net/texmaker/texmakerwin32_install.exe`.

2.3 Ultimi preparativi

Ora che avete tutto il software necessario per poter lavorare, ciò che vi resta da fare è organizzare il lavoro. Scrivere un documento in LATEX, infatti, significa soprattutto creare un progetto. Seguite un paio di consigli e non avrete problemi.

In primo luogo, create una cartella in cui terrete tutti i vostri lavori in LaTeX. All'interno di questa cartella, creerete una subdirectory per ogni specifico progetto.

Ogni documento infatti, in genere si compone di più file (a meno che non si tratti di un documento davvero minimale). Ad esempio per documenti molto complessi solitamente si crea un file per ogni capitolo; inoltre, potreste voler inserire delle figure. In questo caso, tutti i file in uso è bene che stiano nella medesima cartella, non solo per questioni di 'ordine' ma anche per consentire a LaTeX di importare i file giusti.

I file sorgente, cioè quelli contenenti il codice, sono file di testo con estensione *.tex*, che potete creare semplicemente aprendo un file vuoto con l'editor di testo che avete scelto e salvandolo con questa estensione.

2.4 Ora è tutto pronto!

A questo punto, installato LaTeX e l'editor di testo che abbiamo scelto di utilizzare, non ci resta che imparare a comporre un documento.

Le parti fondamentali di un documento LaTeX sono il preambolo e il corpo del testo.

Il preambolo è la parte iniziale del documento, all'interno della quale si scelgono le caratteristiche tipografiche principali che LaTeX deve utilizzare nella realizzazione del documento finito.

In primo luogo, si sceglie la classe, cioè il tipo di documento che vogliamo realizzare: può essere un libro (book), un articolo (article), e tanto altro ancora. Tale scelta si effettua tramite il comando `\documentclass`:

```
\documentclass[12pt,a4paper]{⟨book⟩}
```

La classe scelta per questo documento è la classe *book* e quindi potete vedere direttamente lo stile con cui questa classe compone. La classe scelta va fra parentesi graffe, mentre quelle fra parentesi quadre sono delle opzioni: nel nostro caso, si è scelto un carattere di 12pt di dimensione e una formattazione del testo adatta ad un foglio di carta A4.

Subito dopo la dichiarazione della classe del documento, bisogna indicare i pacchetti che si vogliono usare. Il comando da dare in questo caso è `\usepackage`:

```
\usepackage[italian]{babel}
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}
```

Come vedete, anche con questo comando è possibile scegliere delle opzioni. I tre pacchetti babel, fontenc e inputenc servono a dichiarare la lingua (o le lingue) in cui sarà scritto il documento, in modo che LaTeX utilizzi le corrispondenti regole per andare a capo, dia i nomi giusti a capitoli e paragrafi (Capitolo, e non Chapter, che è quello che otterremmo se non dichiarassimo nulla: LaTeX, di default, parla inglese...); per dichiarare di voler usare un set di caratteri che contenga già le lettere accentate e che il file sorgente è scritto usando i caratteri accentati della tastiera[1].

[1] La codifica *latin1* generalmente è adatta alla maggior parte dei PC in uso in Italia; per le macchine Windows potrebbe essere più adatto specificare la codifica *ansinew*; con i sistemi più moderni potrebbe essere una buona idea quella di usare la codifica *utf8*, ma per questo scopo

ATTENZIONE! Per quanto riguarda la sillabazione il pacchetto babel *non carica il file dalla sillabazione* della lingua esplicitata come opzione, per esempio l'italiano, ma *seleziona solamente* le regole di sillabazione già configurate al momento dell'installazione.
Potete controllare facilmente se la sillabazione per l'italiano, o per qualunque altra lingua che volete usare, sia già stata installata. Basta compilare un documento qualsiasi con **latex** e poi leggere il contento del file `.log` relativo a quel documento. Le prime righe di questo file elencano le lingue già impostate.
Come regola generale si può affermare che se avete installato il sistema dal DVD TEXlive 2009 o successivo, esso in prima installazione dovrebbe essere già configurato per tutta la quarantina di lingue che LATEX riesce a gestire.
MiKTeX, invece, inizialmente è configurato solo per una mezza dozzina di lingue, fra le quali non è compreso l'italiano. È facile usare il pannello *MiKTeX Settings* per marcare le lingue di proprio interesse, e poi MiKTeX si riconfigura da solo per gestirle.
Il vecchio teTeX, tipico delle piattaforme Linux, andava configurato per installare le lingue di proprio interesse, perché di default aveva solo l'inglese.
Attenzione, dunque, se scrivete in italiano, verificate che l'italiano sia una delle lingue installate.

Inoltre il preambolo è il luogo deputato alla definizione di nuovi comandi e nuovi ambienti tramite i comandi `\newcommand` e `\newenvironment`. Queste però sono funzioni avanzate, per le quali potete consultare manuali di LATEX più avanzati (troverete tutti i riferimenti necessari nella bibliografia). Per il momento questo dovrebbe bastare. Man mano che ci si trova di fronte alla necessità di utilizzare comandi specifici di altri pacchetti, è sufficiente aggiungerli al preambolo e continuare a scrivere. Se il pacchetto non fosse presente nella distribuzione di LATEX che abbiamo scaricato, sarà necessario, come abbiamo accennato in precedenza, installarlo. A questo tema dedicheremo una sezione apposita nel capitolo 3, paragrafo 3.1. A questo punto, diamo il comando che segnala la fine del preambolo e l'inizio del documento vero e proprio:

```
\begin{document}
```

Quando avremo finito di comporre il nostro testo, ci basterà scrivere, in fondo a tutto:

```
\end{document}
```

Tutto ciò che verrà scritto dopo tale comando verrà ignorato, quindi state bene attenti!

Subito dopo aver dato inizio al documento, potete cominciare a scrivere. Però, soprattutto se state scrivendo con la classe *book*, potreste desiderare una prima pagina in cui compaia il titolo del vostro lavoro e il nome dell'autore, cioè voi. In questo caso vi basta utilizzare i comandi che seguono:

è necessario usare un editor testuale capace di usare questa codifica. Tuttavia se non c'è la necessità di inserire caratteri di altri alfabeti (greco, cirillico ebraico, arabo, cinese, giapponese, eccetera) le codifiche *latin1* e *ansinew* vanno benissimo.

```
\title{⟨Il titolo del libro⟩}
\author{⟨Autore⟩}
\date{⟨la data che volete⟩}
...
\maketitle
```

Con \title dite a LaTeX qual è il titolo del vostro libro, e questo titolo verrà scritto nel punto in cui avete inserito il comando \maketitle.

Se non specificate una data precisa, LaTeX metterà quella del giorno corrente. Se non volete che la data compaia, vi basterà lasciare vuote le parentesi graffe.

L'ultima cosa che generalmente trovate all'inizio di un libro è l'indice generale (l'indice analitico, se presente, va alla fine). Naturalmente, anche questo si può ottenere con un semplice comando:

```
\tableofcontents
```

L'indice verrà generato automaticamente: ogni volta che inserirete un nuovo capitolo, o un nuovo paragrafo, LaTeX lo aggiornerà. Questa operazione è asincrona, come diverse altre operazioni che richiedono che certe informazioni siano salvate in file di servizio; per cui è necessario compilare il documento ancora una volta affinché sia aggiornata anche la pagina dell'indice.

2.5 Classi di documenti

Prima di procedere oltre, vediamo una breve panoramica di quali sono le classi di documenti disponibili in LaTeX e le varie opzioni che possiamo utilizzare.

2.5.1 Classi principali

Il nome della classe scelta, come sapete, va inserito all'interno di parentesi graffe dopo il comando `\documentclass`.

`article`: è una classe progettata per scrivere articoli – non articoli di giornale, ma su riviste specializzate! – e quindi non prevede l'uso del comando `\chapter`, in quanto la suddivisione è solo in paragrafi e sottoparagrafi.

`book`: serve a scrivere documenti più lunghi e complessi, che contengano una suddivisione in sezioni più articolata e comprensiva di parti e capitoli.

`letter`: è una classe pensata appositamente per la composizione di lettere.

`beamer`: è la classe per le presentazioni – *slides* – di cui però parleremo più diffusamente in seguito, perché ha delle caratteristiche molto particolari.

Ce ne sono molte altre, come *`thesis`* – per le tesi di laurea – e *`report`*, simile a *`book`*, la cui trattazione però va oltre gli scopi di questa guida, che vuole essere introduttiva.

2.5.2 Opzioni principali

Le opzioni vanno inserite tra parentesi quadre dopo `\documentclass` e prima della dichiarazione della classe del documento. Se ne utilizzate più d'una, dovete separarle con delle virgole (niente spazi!).

Detto questo, passiamo in rassegna le principali opzioni:

- *X*pt: dove *X* è la dimensione del carattere del testo normale (10pt se non si specifica altro).
- *aX*paper: dove *aX* è la dimensione del foglio e può assumere i valori 'a4', 'a5', 'letter', 'b5' eccetera.
- twocolumn: ordina a LaTeX di formattare il testo in due colonne.
- titlepage (notitlepage) indica a LaTeX se deve o meno iniziare una nuova pagina dopo il titolo del documento. La classe `article` ha di default la seconda opzione; le classi `book` e `report` la prima.

Esplorate la vostra installazione del sistema TeX, e troverete nella cartella `...\doc` oppure `.../doc`, suddivisa in diverse sotto cartelle, tutta la documentazione ufficiale del sistema, talmente ampia, che si può avere imbarazzo nella scelta; dopo un po' di pratica si impara a distinguere i documenti di primo approccio da quelli molto specializzati. Come primo approccio si suggerisce di consultare nell'ordine `.../doc/latex/base/usrguide.pdf`, `.../doc/latex/base/clsguide.pdf` e `.../doc/latex/base/classes.pdf`. Se non vi bastano le informazioni che trovate sul vostro disco documentatevi su internet, troverete molte altre indicazioni utili che potrete utilizzare nei vostri documenti.

2.6 Caratteri speciali

Come avrete notato dalla lettura del paragrafo precedente, LaTeX utilizza alcuni caratteri speciali:

`^` hat o circonflesso, introduce gli esponenti in modalità matematica;

`{}` parentesi graffe, racchiudono i gruppi;

`%` percento, inizia i commenti;

`$` dollaro, delimita le formule matematiche;

`_` underscore o lineetta bassa, indica i pedici nelle formule matematiche;

`&` ampersand o e-commerciale, funziona da separatore di colonne nelle tabulazioni;

`#` hash o number-sign, indica l'argomento all'interno della definizione di nuovi comandi;

`~` tilde, produce uno spazio insecabile (cioè uno spazio che non può essere separato da quanto precede e quanto segue a causa di un fine riga) o un accento tilde;

`\` backslash o barra rovescia, introduce i comandi.

I caratteri speciali possono essere inseriti in vari modi a seconda della tastiera o del sistema operativo. Con Linux, ad esempio, la tilde è ottenuta con la combinazione `AltGr+^`, mentre sotto Windows tramite la combinazione `Alt+126` (cifre da tastierino numerico!). Lo stesso vale per l'accento grave, che con LaTeX, non volendo fare uso delle lettere già accentate della tastiera, o dovendo apporre questo accento su altre lettere (per esempio sulle vocali maiuscole) deve essere composto con comandi diversi: con Linux `AltGr+'`,mentre con Windows si ottiene con `Alt+096` (cifre da tastierino numerico!).

Per altro ogni buon editor per LaTeX ha i suoi menù che consentono di inserire qualunque segno non accessibile direttamente dalla tastiera.

Dal momento che hanno funzioni specifiche, i caratteri speciali non possono essere ottenuti in stampa semplicemente digitandoli: devono essere preceduti da `\`. Questo sempre ad eccezione di tilde, hat e backslash.

Tilde e hat servono a produrre accenti; per ottenerli in stampa si utilizzano quindi i codici `\~{}` e `\^{}`. Per quanto riguarda backslash, non è possibile farlo precedere da un altro backslash per il semplice fatto che il codice `\\` ha già un altro significato: serve per le interruzioni di riga. Per ottenerlo in stampa si usa pertanto `\textbackslash`.

2.7 Capitoli, paragrafi e sottoparagrafi

Eccoci quindi alla fase di scrittura vera e propria. La prima cosa che dovete imparare è come inserire un capitolo, un paragrafo, un sottoparagrafo, e così via.

Anche in questo caso, la sintassi è semplice:

```
\chapter{⟨Titolo del capitolo⟩}
\section{⟨Titolo del paragrafo⟩}
\subsection{⟨Titolo del sottoparagrafo⟩}
```

iniziano, rispettivamente, un capitolo, un paragrafo, un sottoparagrafo, e così via.

Comunque, se usate **TexMaker**, potete fare tutto questo semplicemente selezionando l'opzione che vi interessa dal menù a tendina (vedi figura 2.1, alla voce LaTeX->Sectioning.

Un'ultima cosa, molto importante: se volete iniziare un nuovo capoverso, dovete lasciare una riga vuota. Se vi limitate semplicemente ad andare a capo, LaTeX interpreterà la cosa come un semplice spazio e quindi non inizierà un nuovo capoverso!

2.8 La compilazione

La compilazione è il passaggio fondamentale da compiere per ottenere il file di output (in formato .dvi, .ps o .pdf a seconda del tipo di compilazione utilizzato). Per eseguire tale passaggio, potete utilizzare i pulsanti presenti nell'editor di testo che state utilizzando – generalmente situati nel menù in alto – o, se usate Linux, potete anche digitare da terminale

```
latex percorsodocumento.tex
```

oppure, meglio ancora, se volete produrre direttamente il file in formato pdf:

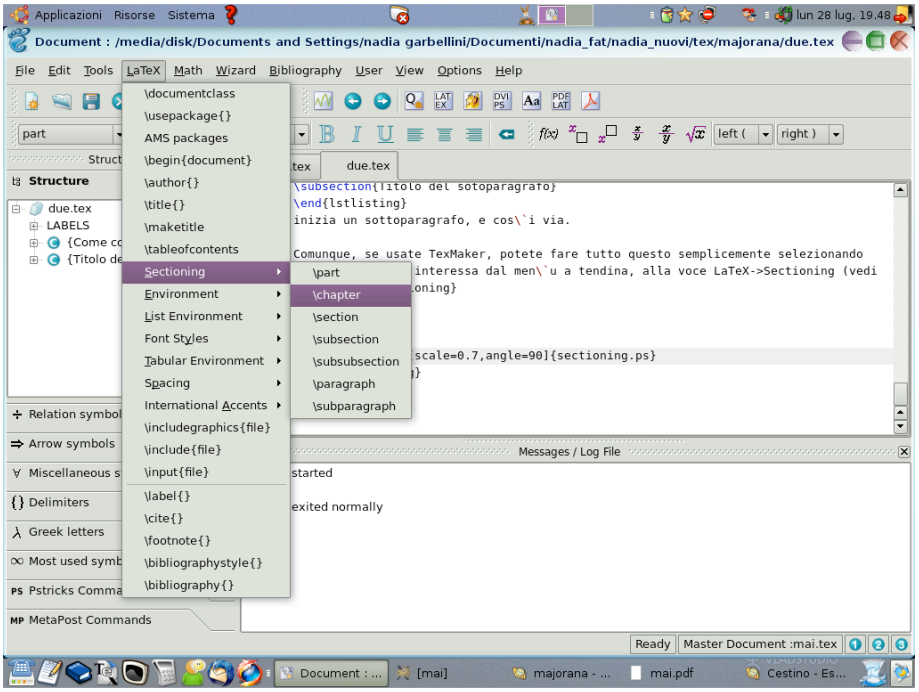

Figura 2.1: Impostare capitoli e paragrafi dal menù di TexMaker

```
pdflatex percorsodocumento.tex
```

Per documenti senza indice, è sufficiente compilare una volta sola. Per documenti con indice, bisogna compilare due volte: con la prima si creano tutti i file ausiliari, e con la seconda l'indice viene inserito nel documento.

Infine, per documenti con indice analitico e/o bibliografia occorrono tre compilazioni: la seconda crea l'indice bibliografico – la bibliografia – e la terza lo inserisce nel documento. Nel caso in cui dovessero esserci degli errori – come dei pacchetti mancanti o dei comandi errati – la compilazione vi restituirà un messaggio di errore, generalmente specificandone il tipo e la riga in cui l'errore stesso è stato fatto. Una volta corretto, bisogna ripetete l'operazione fino a che la compilazione non andrà a buon fine.

2.9 Un piccolo esempio

Vediamo quindi un esempio di documento, veramente minimo, che faccia sfoggio di alcune delle cose viste fin qui. Il codice è:

```
\documentclass [b5paper,12pt,openany]{book}
\pdfpagewidth\paperwidth
\pdfpageheight\paperheight
\usepackage[italian]{babel}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{lmodern}
```

```
\begin{document}
\pagestyle{empty}
\noindent
\begin{minipage}[c][\textheight][c]{\textwidth}\centering
\textsc{\Large Nadia}
\vfill
\textbf{\Huge ESEMPIO}
\vfill
\Large\oldstylenums{2010}
\end{minipage}
\newpage
\begin{minipage}[c][\textheight][c]{\textwidth}
\copyright2010 Nadia

\vspace{50mm}

Stampato in proprio\\
Soggetto alla licenza Creative Commons\\
Finito di stampare il 4 febbraio 2010
\end{minipage}
\newpage
\pagestyle{plain}
\tableofcontents
\chapter{Un piccolo esempio}
\section{Un paragrafo o {\em section}\ldots}
\subsection{\ldots e un sottoparagrafo o {\em subsection}}
\section{Secondo paragrafo}
Con un po' di testo.
\subsection{Come vedete\ldots}
\subsubsection{Un sotto"sottoparagrafo o {\em subsubsection}}
I sotto"sotto"paragrafi non compaiono nell'indice!
\end{document}
```

mentre potete vedere il risultato nella figura 2.2 dove sono rappresentate le quattro pagine del documento: il frontespizio, il retro del frontespizio con le informazioni "legali", l'indice del documento, la prima ed unica pagina del primo capitolo.

Nell'esempio ci sono un paio di novità rispetto a quanto visto finora, dovute al fatto che si è voluto comporre l'esempio direttamente in formato pdf.

- Fra le opzioni si è specificato il formato della carta B5, un formato la cui superficie ha un'area che è la media fra quelle delle carte A4 e A5; lo si è fatto per disporre di un esempio più raccolto come formato, in modo che rimpicciolendolo come si vede nella figura 2.2, pur non potendo leggere i dettagli senza ingrandire la schermata, le parti scritte non risultino troppo piccole. Per altro usando **pdflatex** come motore di composizione, sono stati usati font scalabili, non font bitmap, che non si prestano ad essere ingranditi o rimpiccioliti se non a spese di una visibile granulosità.

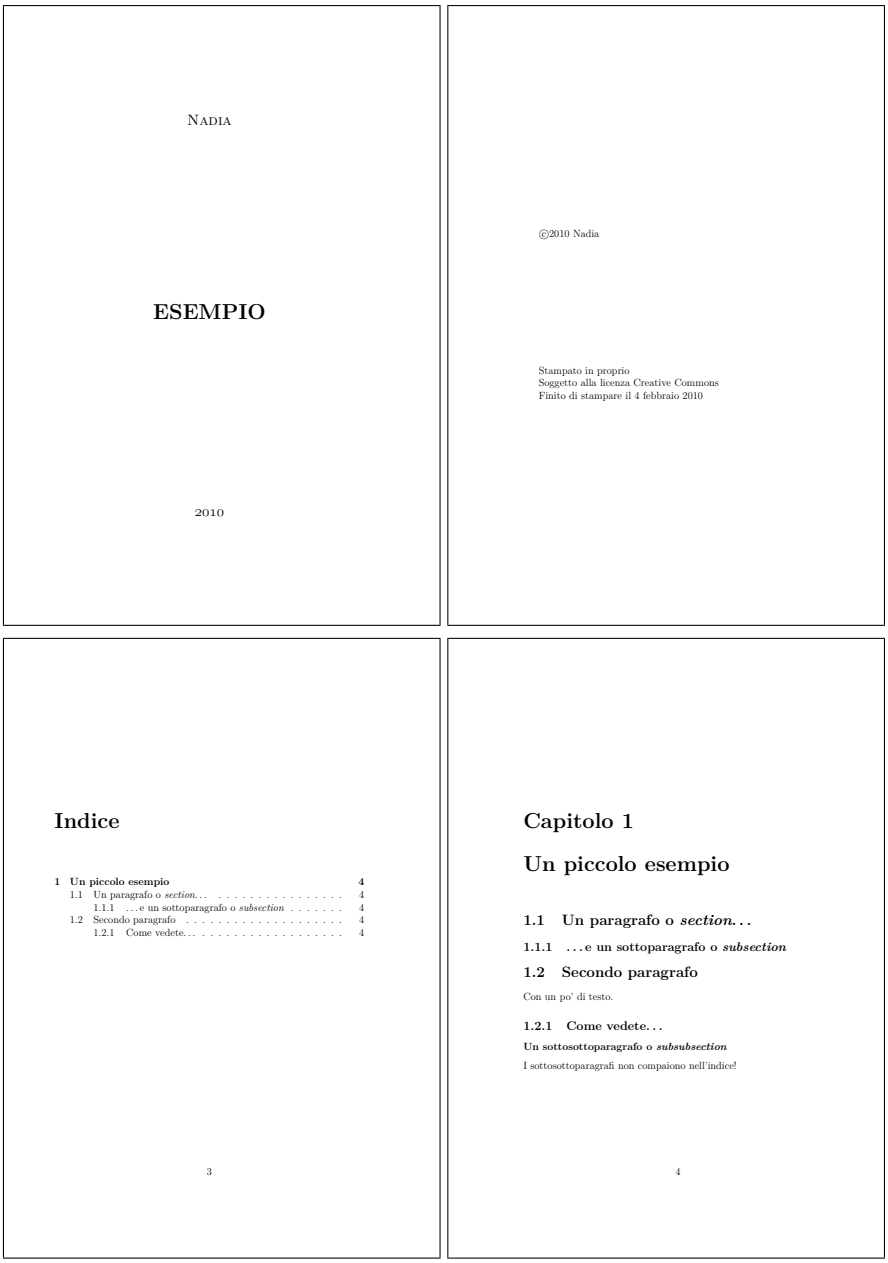

Figura 2.2: Esempio di documento minimo

- Si è usata l'opzione openany per consentire di aprire i capitoli anche sulle pagine di sinistra. Di solito sarebbe una pratica da evitare, specialmente nei libri tecnico scientifici, ma può essere accettata nei testi letterari.

- Avendo specificato un formato di carta diverso da quello di default è stato necessario riassegnare i suoi valori alle variabili `\pgfpagewidth` e `\pgfpageheight`, perché il formato pdf, come anche il formato PostScript,

richiede questa informazione altrimenti non è in grado di posizionare correttamente il testo (lo colloca sul formato di default, che generalmente, al momento dell'installazione del sistema TEX, viene impostato al formato A4). Quando si usa l'opzione a4paper, ovviamente non è necessario fare questa assegnazione.

- Si sono usati font scalabili della collezione Latin Modern mediante l'invocazione del pacchetto lmodern; questi, se non se ne desiderano altri di tipo commerciale, sono il più completo insieme di font compatibili con il sistema TEX; sono consigliabili in ogni circostanza. Se si volessero usare, per esempio, i Times eXtended, basterebbe invocare il pacchetto txfonts; se si volessero usare i Palatino eXtended, basterebbe invocare il pacchetto pxfonts; e via di questo passo per gli altri font distribuiti con il sistema TEX.

- Nel corpo del documento il frontespizio e il suo retro sono stati composti senza usare `\title`, `\author` e `\maketitle`, ma usando certe opzioni dell'ambiente *minipage*, che consentono di specificare sia la larghezza sia l'altezza della minipagina, all'interno della quale si può scegliere come comporre il testo; il frontespizio e il suo retro sono due pagine un po' speciali; per il frontespizio il tipo di composizione eseguito con `\maketitle` è un po' troppo minimalista. Per il retro del frontespizio bisogna di volta in volta scegliere come distribuire le informazioni a seconda di quali informazioni bisogna indicare.

Capitolo 3

Qualcosa in più

Adesso che sappiamo come comporre un documento che contenga il minimo indispensabile, possiamo passare a qualcosa di più elaborato. Vi assicuro che le possibilità offerte da LaTeX sono praticamente illimitate, per cui saremo costretti a passare in rassegna solo le principali. Come ho già avuto modo di dire in precedenza, tuttavia, la rete abbonda di guide, anche molto avanzate, e quindi, quando vi sarete impratichiti, vi potrete sbizzarrire!

3.1 Premessa: pacchetti aggiuntivi

In ambiente Linux Ubuntu, installare pacchetti aggiuntivi è veramente molto semplice. La prima cosa che dovete fare è scaricare l'archivio del pacchetto che vi interessa da CTAN (basta cercare in Google il nome del pacchetto, e troverete immediatamente la pagina che vi interessa. Cercherò comunque di fornirvi tutti i link alle pagine dove trovare i pacchetti di cui parleremo all'interno di questo manuale).

LaTeX ha almeno tre cartelle in cui posizionare i file di stile, quindi i pacchetti:

1. albero principale: contiene i file installati con la distribuzione, disponibili per tutti gli utenti.

2. albero locale: contiene file, disponibili a tutti gli utenti, che si possono modificare senza alterare l'albero principale.

3. albero personale: contiene i file personali, appunto, di ogni singolo utente; questo albero si può modificare inserendo i pacchetti che l'utente utilizza di norma.

Alcune distribuzioni possono avere altre cartelle; la cosa migliore è quella di esplorare il disco fisso.

Per conoscere il percorso di ognuno dei tre alberi suddetti, vi basterà aprire un terminale e digitare, rispettivamente:

```
kpsexpand '$TEXMFDIST'
kpsexpand '$TEXMFLOCAL'
kpsexpand '$TEXMFHOME'
```

Per quanto riguarda l'albero locale, generalmente il suo percorso è `/texmf`. Per aggiungere pacchetti bisogna copiare la relativa cartella nella subdirectory `/texmf/tex/latex`, che potete creare digitando da un terminale:

```
sudo mkdir /texmf/tex/latex
```

La stessa procedura si utilizza per posizionare i pacchetti negli altri due alberi. Tuttavia, in questo caso, una volta copiata la cartella è necessario digitare da terminale:

```
sudo texhash
```

Quest'ultimo comando serve ad aggiornare l'albero delle dipendenze di LaTeX e quindi a fargli riconoscere ed utilizzare il pacchetto appena aggiunto.

In alternativa, si può semplicemente copiare il file di stile nella cartella in cui avete salvato il file *.tex* su cui state lavorando.

Alcuni pacchetti richiedono di essere installati in modo diverso. Se vi dovesse capitare di incontrarne uno, leggete semplicemente la documentazione che vi viene fornita all'interno dell'archivio, che in genere riporta le istruzioni per l'installazione nel caso sia necessario utilizzare delle procedure particolari.

Infine, disponendo della distribuzione TeXlive, si può usare il programma perl **tlmgr**, eventualmente in modalità grafica (talvolta sono necessarie le prerogative dell'amministratore).

Per Mac OS X di solito non ci sono pacchetti da installare, perché l'installazione con MacTeX è completa; ma se, per esempio, vi create pacchetti personali, allora è bene che li carichiate nel vostro albero personale come detto per i sistemi Linux, senza la necessità di usare le prerogative dell'amministratore, e senza la necessità di aggiornare il data base dei nomi dei file, perché l'albero personale è radicato in `Library` è quindi è sempre automaticamente aggiornato.

Con Windows, MiKTeX riconosce se state usando solo i pacchetti già installati; se non fosse così, vi chiede se volete installare i pacchetti di volta in volta mancanti e, alla vostra risposta affermativa, provvede da solo a tutte le operazioni necessarie. Se però aggiungete vostri pacchetti personali all'albero personale, ricordatevi di aggiornare il data base dei nomi dei file.

3.2 Espressioni matematiche

LaTeX, come abbiamo già avuto modo di dire, è particolarmente 'dotato' per scrivere formule matematiche, la cui resa grafica è ottima. Potete scrivere qualunque tipo di formula, dalla più semplice alla più complessa, in maniera estremamente semplice. Naturalmente, anche in questo caso dovete utilizzare dei comandi specifici, che però trovate anche nel menù 'Math', per quanto riguarda le funzioni e gli operatori matematici, e nel menù laterale di TexMaker (vedere figura 3.1) per quanto riguarda i simboli. Ricordate che ci sono vari tipi di 'ambienti' matematici:

1. Brevi formule matematiche all'interno di una linea ('in corpo'), ad esempio:

La formula per ottenere l'area del quadrato di lato l è $A = l^2$.	`La formula per ottenere l'area del quadrato di lato $l$ è $A = l^2 $.`

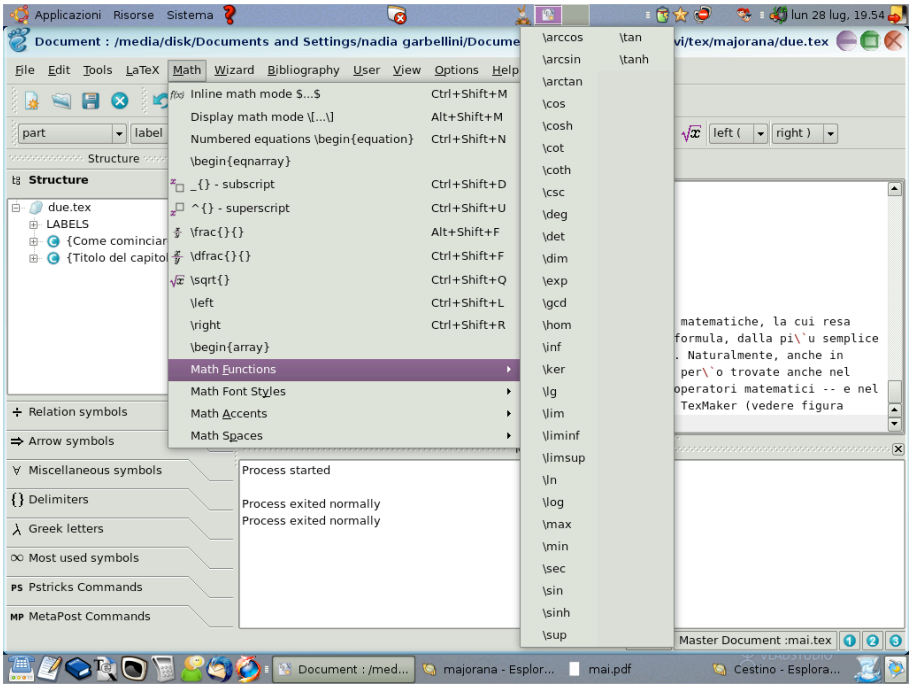

Figura 3.1: Scrivere espressioni matematiche usando il menù 'Math' di TexMaker

Quindi, prima e dopo la formula bisogna inserire il simbolo '$', in modo da far entrare LaTeX in modalità matematica.

2. Formule 'fuori corpo':

$$f(x) = x^2 + \frac{2\sqrt{x}}{34} + \frac{25}{x}$$

```
\[f(x) = x^2 +
\frac{2\sqrt{x}}{34} +
\frac{25}{x}\]
```

In questo caso c'è una distinzione da fare. Se volete che la formula non sia numerata, dovete racchiudere la formula dentro il gruppo di comandi `\[...\]`. Se invece volete che sia numerata, racchiudetela fra `\begin{equation}` e `\end{equation}`.

Se volete essere in grado di fare riferimento, più avanti, alla formula utilizzando la numerazione che le è stata assegnata, aggiungete, subito prima di `\end{equation}` e senza righe vuote, il comando `\label{`*nomeformula*`}`.

Per richiamarla utilizzate il comando[1] `\eqref{`*nomeformula*`}`: il numero verrà messo fra parentesi. Se non le volete, al posto di `\eqref` utilizzate semplicemente `\ref`.

In questo modo, LaTeX gestisce autonomamente tutti i riferimenti incrociati. Se decideste di inserire un'altra equazione prima di quella richiamata, la

[1] Il comando `\eqref` è disponibile solo se si è invocato il pacchetto amsmath.

numerazione si aggiornerebbe automaticamente così come tutti i riferimenti incrociati.

Si osservi che il comando per l'esponente, `^`, come qualunque altro comando, richiederebbe che il suo argomento fosse racchiuso fra parentesi graffe. Se però il suo argomento è costituito da un solo oggetto (una lettera, una cifra, un altro comando senza argomenti...), le graffe possono essere omesse.

Ora vediamo un esempio (scritto a caso, senza alcun significato matematico!) di ciò che potete fare con l'ambiente matematico di LaTeX:

$$\underbrace{2\sin x - x_{\lambda}^{\gamma\delta}}_{prima\ parte} + \underbrace{\int_0^{\infty} \log x^3 + \arctan x_{\omega}}_{seconda\ parte}$$

```
\[
\underbrace{2\sin x-x
_{\lambda}^{\gamma\delta}}
_{prima\ parte} +
\underbrace{\int_0^{\infty}
\log x^3+ \arctan x_{\omega}}
_{seconda\ parte}
\]
```

Leggendo il codice, vedete come si creano esponenti e pedici. In entrambi i casi, le parentesi graffe sono necessarie solo se l'argomento è composto da più di un carattere, altrimenti potete anche non inserirle.

3.3 Ambienti

Gli ambienti, in LaTeX, sono particolari porzioni di testo racchiuse tra i comandi `\begin{⟨nomeambiente⟩}` e `\end{⟨nomeambiente⟩}`. Ne abbiamo già incontrati alcuni (*equation* ad esempio). Vediamo ora quali sono i principali.

3.3.1 Elenchi puntati e numerati

Per quanto riguarda gli elenchi puntati e numerati, i due ambienti principali sono *enumerate* ed *itemize*. La differenza risiede nel fatto che il primo crea un elenco numerato, il secondo un elenco puntato. In entrambi i casi, ogni nuovo punto dell'elenco viene introdotto dal comando `\item`. Ad esempio:

1. primo
2. secondo
3. ...

```
\begin{enumerate}
\item primo
\item secondo
\item \dots
\end{enumerate}
```

- primo
- secondo
- ...

```
\begin{itemize}
\item primo
\item secondo
\item \dots
\end{itemize}
```

Per quanto riguarda l'ambiente *itemize* (lo si potrebbe fare anche on l'ambiente *enumerate*, ma sarebbe sprecato), può essere facilmente personalizzato come segue:

& primo
$ secondo
# ...
abc insomma come volete

```
\begin{itemize}
\item[\&] primo
\item[\$] secondo
\item[\#] \dots
\item[abc] insomma come
volete
\end{itemize}
```

3.3.2 Tabelle

L'ambiente più utilizzato nella creazione di tabelle è *tabular*. Come abbiamo già avuto modo di dire parlando dei caratteri speciali,

Vediamo subito un esempio, che ci aiuterà nella comprensione del funzionamento di questo ambiente:

Titolo tabella		
2 colonne		3
1	2 colonne	
1	2	3
a	b	c

```
\begin{tabular}{ccc}
\hline
\multicolumn{3}{c}{Titolo
tabella}\\
\hline
\multicolumn{2}{c}{2
colonne}& 3\\
\hline
1 &\multicolumn{2}{c}{2
colonne}\\
\hline
1& 2& 3\\
a& b& c\\
\hline
\end{tabular}
```

Cominciamo dalla prima linea di codice: `{ccc}` significa: 'voglio che la mia tabella sia composta da tre colonne, in ciascuna delle quali il testo deve essere posizionato al centro'. Quindi, subito dopo aver dichiarato l'inizio della tabella, occorre specificare, per ogni colonna, come vogliamo che il testo venga composto: al centro ('c': centre), a destra ('r': right) o a sinistra ('l': left). Se avessimo voluto quattro colonne, di cui la prima e l'ultima con il testo composto, rispettivamente, e sinistra e a destra e le due centrali con il testo, appunto, centrato, avremmo dovuto scrivere: `{lccr}`.

Passiamo ora alla seconda linea: con il comando `\hline` tracciamo una linea orizzontale (horizontal line).

Il comando `\multicolumn` serve a scrivere del testo disposto in modo tale da occupare più colonne. In questo caso il testo 'Titolo tabella' occupa tre colonne ed è centrato.

Come abbiamo già avuto modo di dire parlando dei caratteri speciali, il carattere `&` serve a dividere le colonne all'interno delle tabulazioni.

Inoltre, alla fine di ogni colonna bisogna aggiungere `\\`, comando con il quale dichiariamo la fine di ogni riga della nostra tabella.

Naturalmente ci sono molte altre cose che si possono fare con questo ambiente, come scrivere testo su più righe, colorare le celle, ecc. Anche in questo caso, però, si tratta di comandi avanzati per i quali rimandiamo a guide più complete.

3.4 Inserire figure e tabelle

Abbiamo già visto come creare delle tabelle.

Tuttavia, per fare in modo che tabelle e immagini, siano numerate, abbiano una didascalia e possano essere richiamate in seguito dobbiamo utilizzare due ambienti particolari: *table* e *figure*. Per spiegare nella maniera più semplice possibile di che cosa si tratti, diciamo che questi due ambienti sono delle 'scatole' dentro le quali inserire tabelle e immagini a nostro piacimento. LaTeX le riconosce e le utilizza per la gestione di numerazioni, indici, riferimenti e quant'altro.

Ambiente *table*

```
\begin{table}
Tabella creata usando
tabular
\caption{didascalia}
\label{nome}
\end{table}
```

Ambiente *figure*

```
\begin{figure}
Figura inserita con:
\includegraphics{file
grafico}
\caption{didascalia}
\label{nome}
\end{figure}
```

Come vedete, le etichette si assegnano come per le equazioni. Qui in più abbiamo le didascalie (*caption*); attenzione: il comando `\label` con il suo argomento deve essere messo *dopo* il comando `\caption` e il suo argomento.

Per quanto riguarda le figure, come vedete, si inseriscono creando la 'scatola' appropriata e poi inserendo il nome (o il percorso se non si trova nella cartella in cui stiamo salvando tutto il nostro lavoro) del file grafico che contiene la nostra immagine. Qui occorre stare attenti: se stiamo compilando utilizzando come compilatore il programma **latex**, allora le immagini andranno inserite nei formati *.ps* o, meglio, *.eps*. Se invece usiamo il programma **pdflatex**, dovremo utilizzare i file di immagini cui siamo abituati, come *.png*, *.jpg*, o, meglio ancora, *.pdf*, (oppure *.mps*, ma questo formato è conosciuto da pochi). Il comando `\includegraphics` sa da solo quali formati può includere nel documento da comporre, quindi è *inopportuno* specificare l'estensione del file grafico.

Di default **latex** accetta solo file PostScript, possibilmente del tipo 'encapsulated' con l'estensione *.eps*, mentre **pdflatex** accetta solo altri formati: *.png*, *.jpg*, *.pdf*. I due set di formati non sono mutuamente compatibili, quindi bisogna provvedere alle debite conversioni. Vale la pena di ricordare il programma **jpegtops** per trasformare i formati fotografici in formato 'encapsulated PostScript' (lo si specifica esplicitando l'estensione del file di uscita); alternativamente si può ricorrere al programma **ps2pdf** (generalmente fornito insieme al sistema TeX) che trasforma i file PostScript, anche gli 'encapsulated', in file *.pdf*.

È opportuno ricordare di non usare i formati sbagliati in relazione al tipo di immagine da includere nel documento. Per i disegni al tratto (grafici, schemi, schizzi, ecc.) sono più adatti i formati vettoriali *.pdf* e *.eps*, mentre il formato *.jpg* è più adatto alle fotografie e alle immagini con colori sfumati. Se proprio non se ne può fare a meno, per esempio quando si usa uno scanner per scandire un diagramma stampato, è preferibile usare il formato *.png* piuttosto che il

formato *.jpg*, perché i disegni al tratto in quest'ultimo formato presentano degli artefatti piuttosto fastidiosi; questo fenomeno è dovuto al tipo di compressione dei dati dell'immagine. Non è il caso di scendere nei dettagli; questi non sono difetti di LaTeX, ma sono comuni a qualunque programma di composizione si usi. Gli editori (stampatori) più esigenti esigono figure vettoriali o figure a matrici di pixel con una densità minima di 300 punti al pollice. Sullo schermo questi difetti non si vedono perché gli schermi hanno densità di pixel generalmente non superiori a 100 pixel al pollice.

Questo è molto importante: se cercate di compilare un sorgente contenente immagini in formati non 'riconoscibili' con il particolare tipo di compilatore che state utilizzando, otterrete un errore e molto probabilmente non sarete nemmeno in grado di creare un file di output.

3.4.1 Figure e testo riquadrati

Un'ulteriore possibilità che LaTeX vi fornisce è quella di inserire figure e testo in una 'scatola' (box), incorniciata da un riquadro ed eventualmente colorata. Il comando adatto allo scopo è `\framebox`, oppure `\fbox`. Il primo, a differenza del secondo, ci permette di utilizzare delle opzioni. Vediamo qualche esempio del funzionamento di tali pacchetti.

ciao
```
\fbox{ciao}
```

ciao
```
\framebox{ciao}
```

ciao
```
\framebox[4cm]{ciao}
```

ciao
```
\framebox[3cm]{ciao}
```

ciao
```
\framebox{\colorbox{yellow}{ciao}}
```

ciao
ciao
```
\framebox[4cm]{%
\begin{minipage}{35mm}
\centering
ciao\\
ciao
\end{minipage}}
```

Come vedete, `\fbox{`⟨*testo*⟩`}` permette di riquadrare solo parti di testo lunghe non più di una riga. Lo stesso vale per `\framebox{`⟨*testo*⟩`}`, a meno che non utilizziamo degli 'espedienti' per poter introdurre più righe di testo.

Uno di questi consiste nell'utilizzo dell'ambiente *minipage* all'interno di `\framebox`, che richiede di specificare la larghezza sia del riquadro (tra parentesi quadre) oltre che della *minipage* stessa (tra parentesi graffe). Quest'ultima può essere minore della prima, se vogliamo un po' di spazio ai lati. Aggiungiamo che per andare a capo dobbiamo aggiungere `\\` alla fine della riga, proprio come per le tabelle.

Arriviamo così al secondo espediente che possiamo utilizzare, cioè quello di utilizzare l'ambiente *tabular* all'interno di `\framebox`, per ottenere lo stesso risultato.

All'inizio tutto questo può sembrare molto complesso, ma vi assicuro che un po' di esperienza – e di pazienza – permette di ottenere ottimi risultati in poco tempo.

3.5 Gestire documenti complessi

Uno degli usi più diffusi di LaTeX fra gli studenti è probabilmente la stesura di tesine e tesi di laurea, che spesso sono formate da molti capitoli. Quando ci si trova di fronte a documenti di questo genere, risulta molto utile disporre di una struttura portante, il cosiddetto *master document*, sul quale innestare i vari capitoli. Con LaTeX questo risultato è molto semplice da ottenere.

In primo luogo, creiamo un documento, che chiameremo ***master.tex***, in cui inseriremo il preambolo, quindi tutti i pacchetti che ci possono servire, e gli eventuali nuovi comandi che riteniamo possano servirci.

In secondo luogo, creiamo altri documenti, tutti con estensione ***.tex***, uno per ogni capitolo del nostro lavoro. Possiamo chiamarli ***capuno.tex***, ***capdue.tex***, e così via. Sarebbe opportuno che tanto il master document quanto i singoli file dei vari capitoli avessero dei nomi un po' più mnemonici in relazione al loro contenuto. Questo ripara da numerosi errori, anche se può sembrare una cosa banale; pensate a quando eseguite la ricerca di un file che contenga una stringa di testo specificata; che risultato sarebbe se il vostro programma di ricerca di indicasse dodici file, tutti con lo stesso nome ***master.tex***. Certo potete cambiare stringa di ricerca, ma non è questo il punto; ogni lavoro di composizione, ogni testo che componete, oltre ad essere collocato in una cartella con un nome significativo, dovrebbe avere tutti i suoi file componenti con nomi significativi e assolutamente diversi da quelli degli altri documenti.

A questo punto riprendiamo il nostro master document e includiamo i vari capitoli utilizzando il comando `\include{`⟨*nomefile*⟩`}` (⟨*nomefile*⟩ senza estensione!). Nella figura 3.2 vediamo un esempio di master document.

Noterete alcune novità rispetto a quanto detto fin qui.

Il comando nuovo `\includeonly{`⟨*listadifile*⟩`}` serve per dire che, nonostante dentro al corpo del documento si ordini di inserire un certo numero di file corrispondenti a vari capitoli, appendici, eccetera, in realtà si vogliono includere solo quelli esplicitamente elencati nella ⟨*listadifile*⟩, ma senza perdere le informazioni degli altri file, se sono già disponibili, senza perdere tempo a ricompilare tutto quanto. È molto comodo durante la lavorazione di un documento complesso poter compilare un capitolo alla volta. Tutti i programmi coinvolti rispondono più rapidamente. Nello stesso tempo è bene che le informazioni relative ai capitoli già compilati restino disponibili, in modo da risolvere via via i riferimenti incrociati che permettono attraverso il comando `\ref` di accedere ai numeri delle equazioni, delle figure, delle tabelle, e di quant'altro sia numerabile e a cui ci si voglia riferire. Va da sé che la versione completa del documento va compilata alla fine senza fare ricorso al comando `\includeonly`.

Il comando `\bibliographystyle{`⟨*stile*⟩`}`. Con questo comando si dichiara lo stile della bibliografia. Noi abbiamo scelto lo stile Harvard, di cui notate in

```
\documentclass[12pt,a4paper]{book}

\usepackage[italian]{babel}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{amssymb,amsmath,multicol,tabularx,makeidx}
\usepackage{fancyhdr,graphicx,color,listings}
\usepackage[dcucite]{harward}
\bibliographystyle{harward}
\usepackage{pstricks,pstricks-add}
\usepackage{pst-plot}
\usepackage{pst-node}
\usepackage{hyperref}

\includeonly{capuno,capdue}

\begin{document}
\title{\LaTeX}

\maketitle

\tableofcontents

\include{capuno}
\include{capdue}
\include{capconclusioni}
\bibliography{bibliografia}

\end{document}
```

Figura 3.2: Un esempio di master document

preambolo il corrispondente pacchetto, ma ce ne sono moltissimi altri, ed esiste anche la possibilità di creare degli stili bibliografici propri (si tratta ovviamente di una funzione avanzata, di cui non è il caso di parlare in questa sede).

In secondo luogo, il comando `\bibliography{`⟨*bibliografia*⟩`}`. Questo comando crea la bibliografia utilizzando un file bibliografico da noi creato, con estensione *.bib* – in questo caso il file sarà *bibliografia.bib* – e salvato nella nostra directory di lavoro. All'interno di questo file scriveremo i riferimenti bibliografici a tutti i libri, articoli, riviste, siti web, eccetera, che abbiamo citato nel corso del nostro lavoro.

IMPORTANTE: si devono sempre citare le fonti che utilizziamo nel nostro lavoro. Non farlo significa appropriarsi indebitamente di qualcosa che nostro non è. Se questo discorso vi sembra contrario alla filosofia open source, vi sbagliate: gli autori di software e documentazione libera non vi impediscono di utilizzare il frutto del loro lavoro, né vi impongono di pagare per farlo. Tuttavia, si tratta sempre di qualcosa che è stato da loro pensato, studiato e realizzato: citarli significa semplicemente attribuire loro la paternità di quanto hanno fatto.

```
% file bibliografico:  bibliografia.bib
@Article{etichettaart,
author = {⟨Autore articolo⟩},
title = {⟨Titolo articolo⟩},
journal = {⟨Nome rivista⟩},
year = {⟨Anno pubblicazione⟩},
OPTvolume = {},
OPTnumber = {},
OPTpages = {},
}
@Book{etichettabk,
author = {⟨Autore libro⟩},
editor = {⟨Curatore edizione⟩},
title = {⟨Titolo libro⟩},
publisher = {⟨Casa editrice⟩},
address ={⟨Città o indirizzo più esteso sede della casa editirice⟩},
year = {⟨Anno pubblicazione⟩},
}
```

Figura 3.3: Esempio di file di bibliografia: *bibliografia.bib*

Chiusa questa piccola raccomandazione, torniamo alla spiegazione. Ogni editor di testo, o quasi, ha un'apposita sezione del menù dedicata alla bibliografia. Con TexMaker, in alto potete vedere la voce Bibliography, da cui potete selezionare il tipo di fonte che state utilizzando (libro, articolo, intervento tenuto ad una conferenza, ecc.).

Naturalmente, dovete prima di tutto aver creato il file di bibliografia. Da lì utilizzate il sopra citato menù: selezionando una voce, vi compaiono i relativi campi: titolo, autore, editore, curatore,[2] eccetera. Compilateli con le relative informazioni, e LaTeX penserà a produrre la bibliografia.

La figura 3.3 vi mostra un piccolo esempio di file bibliografico. L'etichetta è quella che richiameremo nel testo quando avremo bisogno di citare una fonte, usando il comando `\cite{⟨etichetta⟩}`. I campi preceduti dalla stringa OPT sono campi opzionali – gli altri sono obbligatori – e, nel caso in cui dovessimo decidere di riempirli, dobbiamo cancellare le tre lettere'OPT'. Una volta terminata la produzione del file, dovete usare una precisa procedura di compilazione: una volta **latex**, una volta **bibtex**, tre volte di nuovo **latex**.

Tornando per un momento allo stile della bibliografia, bisogna sottolineare che la maggior parte di quelli preesistenti crea bibliografie in inglese. Tuttavia, esistono siti internet dove trovare degli stili in italiano, per esempio `http://www.guit.sssup.it/`, (a meno che non vogliate cimentarvi nell'impresa di crearne uno tutto vostro!).

[2] I campi sono in inglese. In inglese editor significa curatore, mentre la parola italiana 'editore' si traduce con *publisher*. Attenzione a non fare confusione!

Capitolo 4

La grafica

LaTeX consente di eseguire dei disegni al tratto in diversi modi. D'accordo, LaTeX è un compositore/impaginatore di documenti, è un *text processor*, non è uno strumento di grafica. Tuttavia i documenti tecnico scientifici richiedono molto spesso di presentare dei disegni relativamente semplici, diagrammi cartesiani a due o a tre dimensioni, schemi a blocchi, statistiche espresse con diagrammi a torta o mediante istogrammi.

Certo esistono altri programmi esplicitamente dedicati a questo tipo di disegni, ma presentano spesso alcuni inconvenienti, al di là delle loro specifiche funzionalità. Lo svantaggio tipograficamente più rilevante è che le parti scritte, le legende interne, le etichette che identificano le parti del disegno vengono di solito composte con caratteri diversi da quelli usati nel testo. Ciò è brutto e balza all'occhio più di quanto non si creda.

Per questo, nei limiti del possibile, è desiderabile eseguire i disegni di questo tipo direttamente mediante lo stesso programma che compone e impagina il testo.

4.1 I programmi di grafica del sistema TeX

Esistono sostanzialmente tre livelli di funzionalità a cui si può accedere.

1. Si può usare l'ambiente *picture* nativo di LaTeX così come descritto nella Guida di Leslie Lamport del 1994, anno di nascita dell'attuale versione LaTeX 2_ε del linguaggio di mark-up di cui si occupa questa guida, e cioè mediante l'uso del pacchetto pict2e di data successiva o uguale al 2009/01/01.

2. Si può usare il pacchetto PSTricks formato da numerosi file, che vanno richiamati solo per produrre il tipo di disegno che interessa. Questo pacchetto si rifà al linguaggio PostScript, quindi può essere usato solo con il compilatore **latex**, *non* con **pdflatex**. Se interessa il prodotto finale in formato pdf, occorre trasformare il file di uscita, che è nel formato nativo del sistema TeX, che ha estensione *.dvi* (DeVice Independent), per poi trasformarlo nel formato pdf attraverso l'uso di programmi accessori del sistema TeX, segnatamente **dvipdfm**, che esegue la trasformazione diretta, oppure **dvips** che trasforma il file *.dvi* nel formato PostScript con

estensione *.ps*, e poi con **ps2pdf** che trasforma il file PostScript nel file finale *.pdf*. Generalmente queste trasformazioni possono essere eseguite con un semplice click del mouse su appositi 'bottoni' nella barra degli strumenti dell'editor stesso.

3. Si può usare il pacchetto pgf con la sua appendice tikz. L'acronimo PGF significa 'portable graphic format', e comprende una serie di macro che permettono di eseguire disegni, come anche succede usando PSTricks, inserendo nel file di uscita gli operatori e i loro argomenti scritti direttamente nel linguaggio del programma che dovrà poi provvedere alla stampa o alla visualizzazione del documento; saranno dunque codice PostScript se si sta componendo con **latex**, mentre si tratterà di codice pdf se si sta componendo con **pdflatex**. Ti*k*Z è un acronimo che deriva dal tedesco e indica una serie di file che formano il pacchetto tikz; essi definiscono l'ambiente di disegno *tikzpicture* dentro al quale sono usabili le varie macro che generano il codice di disegno adatto al particolare tipo di disegno che si sta eseguendo.

Non si scenderà nei dettagli per la seconda e la terza soluzione; essi sono argomenti abbastanza avanzati che conviene approfondire nelle rispettive guide, reperibili, come le altre guide, nel vostro stesso disco fisso in `.../doc/generic/pstricks-tutorial/` (in vari file che occupano un capitolo ciascuno) e in `.../doc/generic/pgf/pgfmanual.pdf`.

Vale la pena, però, di sottolineare che le guide citate sono molto istruttive e molto complete; i disegni che si possono ottenere sono eccezionali con entrambi i metodi. Sebbene pgf offra ancora solo il 90% della funzionalità di pstricks, ci sono lavori che possono essere eseguiti solo con quest'ultimo pacchetto; altrimenti in generale è più facile servirsi del primo, anche se la moltitudine di macro che bisogna imparare ad usare è rilevante in entrambi i casi; d'altra parte con questi pacchetti si vogliono fare disegni complessi, quindi...

4.2 L'ambiente *picture*

Fin dalla sua nascita nel 1984 LaTeX disponeva di un ambiente *picture* che consentiva di fare dei modesti disegni, ma disponeva di caratteristiche particolari che lo rendevano prezioso anche in applicazioni che nulla avevano a che vedere con il disegno. In quell'ambiente c'erano notevoli limitazioni grafiche; con la nascita di LaTeX 2_ε nel 1994 Leslie Lamport ha annunciato un pacchetto di estensione per rimuovere quelle limitazioni. Finalmente nel 2003 è stato messo a disposizione degli utenti il pacchetto pict2e che eliminava le limitazioni dell'ambiente originario; questo pacchetto è diventato stabile nel 2004 e nel 2009 esso è stato ulteriormente esteso al punto di non avere nulla da invidiare a pacchetti più elaborati di disegno. Ora quello che gli altri pacchetti hanno in più, rispetto a quanto mette a disposizione pict2e, sono certi automatismi di elaborazione che rendono il disegno più comodo da eseguire.

Questo ambiente riporta la grandezze metriche ad un unica unità di misura specificabile con `\unitelength=`⟨*unità di misura*⟩, che non va più ripetuta né cambiata all'interno dell'ambiente, ma deve essere specificata solo prima dell'apertura dell'ambiente stesso. Come unità di misura si possono usare sia quelle che LaTeX conosce: mm (millimetro), cm (centimetro), in (pollice anglosassone=

25,4 mm), pt (punto tipografico americano = 1 in/72,27), bp (punto tipografico PostScript = 1 in/72), dd (punto tipografico europeo=0,376065 mm), pc (pica=12 pt), cc (cicero=12 dd). LaTeX capisce anche le unità di misura relative alla grandezza (o corpo) del font corrente: ex (altezza della lettera 'x'), em (larghezza della lettera 'M'); inoltre si possono usare come riferimento anche multipli o sottomultipli di altre dimensioni, per esempio si potrebbe prendere come unità di misura "un centesimo della larghezza del testo" scrivendo:

```
\unitlength=0.01\textwidth
```

L'ambiente *picture* richiede che in fase di apertura siano dichiarate le sue dimensioni reali o apparenti:

```
\begin{picture}(⟨base⟩,⟨altezza⟩)(⟨deltax⟩,⟨deltay⟩)
⟨istruzioni per il disegno⟩
\end{picture}
```

dove ⟨*base*⟩ e ⟨*altezza*⟩ sono la base e l'altezza del rettangolo vero o apparente che racchiude il disegno. Se si desidera che le coordinate dei punti da collocare nel disegno siano riferite ad assi cartesiani la cui origine *non* coincide con lo spigolo inferire sinistro del suddetto rettangolo, si specificano mediante ⟨*deltax*⟩ e ⟨*deltay*⟩, le coordinate dello spigolo inferiore sinistro rispetto all'origine degli assi cartesiani. Se questo spostamento non deve avere luogo, si omette completamente il secondo paio di parentesi tonde e il loro contenuto. Si noti che i numeri decimali fratti si scrivono con il *punto* decimale, perché la virgola viene usata per separare i due valori delle coordinate.

Si è sottolineato il fatto che le dimensioni possono essere vere o apparenti. Non credo che ci vogliano spiegazioni per 'vere', ma bisogna spiegare bene che cosa vuol dire 'apparenti': infatti la scatola ideale che contiene effettivamente il disegno ha certe dimensioni, ma quelle che si specificano con ⟨*base*⟩ e ⟨*altezza*⟩ sono quelle che noi *facciamo credere* a LaTeX che siano le dimensioni del disegno. Al limite possiamo far credere a LaTeX che il disegno abbia dimensioni nulle sia in orizzontale sia in verticale; questo viene sfruttato da diversi pacchetti del sistema TeX, ma può essere usato anche da noi, per collocare oggetti sulla pagina in posizioni qualsiasi, componendo il testo come se quegli oggetti non ci fossero.

Dentro all'ambiente originale, tutti gli oggetti vengono collocati in posizione mediante il comando `\put`, oppure una sequenza di oggetti uniformemente distanziati, mediante il comando `\multiput`. Nell'ambiente ridefinito mediante il pacchetto pict2e ci sono anche dei comandi di disegno che collocano i loro segni nella posizione corretta anche senza usare `\put` e `\multiput`.

La sintassi di questi due comandi è:

```
\put(⟨x⟩,⟨y⟩){⟨oggetto⟩}
\multiput(⟨x⟩,⟨y⟩)(⟨dx⟩,⟨dy⟩){⟨N⟩}{⟨oggetto⟩}
```

dove⟨*oggetto*⟩ è uno dei segni grafici di cui si dirà fra poco; ⟨*x*⟩,⟨*y*⟩ sono l'ascissa e l'ordinata del punto nel quale si vuole collocare l'⟨*oggetto*⟩; più precisamente sono l'ascissa e l'ordinata del punto che si vuol far coincidere con il punto di riferimento dell'⟨*oggetto*⟩; questo punto di riferimento, solitamente è lo spigolo inferiore sinistro del rettangolo circoscritto, ma ci sono certi oggetti, come i cerchi, che hanno il punto di riferimento al loro centro.

Per il comando `\multiput` ci sono ancora altre informazioni: ⟨*dx*⟩,⟨*dy*⟩ sono lo spostamento orizzontale e verticale con cui ogni oggetto viene ripetuto rispetto

all'oggetto precedente, e ⟨*N*⟩ è un numero intero che dice quanti oggetti in totale `\multiput` deve mettere in posizione; si veda il codice usato per creare la griglia di un diagramma cartesiano nella figura 4.6. Gli oggetti possono essere traslati solo lungo linee rette, perché i valori di ⟨*dx*⟩,⟨*dy*⟩ non cambiano di iterazione in iterazione.

L'⟨*oggetto*⟩ in ogni caso può essere del testo o delle immagini importate con il comando `\includegraphics` già visto per le figure. Ma in generale è preferibile, anche per questi particolari oggetti, metterli in posizione come se fossero inclusi in alcuni dei prossimi comandi.

Gli oggetti che richiedono di essere messi in posizione con `\put` o `\multiput` sono i seguenti:

`\line(`⟨*x-pend*⟩`,`⟨*y-pend*⟩`){`⟨*lunghezza*⟩`}` È un segmento rettilineo, di pendenza specificata mediante ⟨*x-pend*⟩,⟨*y-pend*⟩ e la cui proiezione sull'orizzontale vale ⟨*lunghezza*⟩; se e solo se ⟨*x-pend*⟩ è nulla (segmento verticale), la ⟨*lunghezza*⟩ si riferisce alla componente verticale del segmento. Il punto di riferimento è l'estremo del segmento che viene collocato dal comando `\put`, e il segmento si estende nella direzione specificata dai coefficienti di pendenza. Inizialmente questi coefficienti di pendenza potevano essere solo interi non superiori a 6, e i due valori dovevano essere primi fra di loro; ora con l'estensione pict2e possono essere numeri reali qualsiasi, inferiori, però, in valore assoluto a 16 384. Questo vale anche per i coefficienti di pendenza dei vettori.

`\vector(`⟨*x-pend*⟩`,`⟨*y-pend*⟩`){`⟨*lunghezza*⟩`}` È un vettore che ha per punto di riferimento l'estremità della coda, quindi punta nella direzione specificata dai coefficienti di direzione. Se la lunghezza venisse specificata di valore zero, la punta della freccia verrebbe disegnata lo stesso e nella direzione giusta, con il vertice della punta che coincide con il punto di riferimento.

`\circle{`⟨*diametro*⟩`}` È una circonferenza del ⟨*diametro*⟩ specificato. Il punto di riferimento è il suo centro.

`\circle*{`⟨*diametro*⟩`}` È un disco circolare pieno, non solo la sua circonferenza; per il resto vale quanto detto per `\circle`.

`\makebox(`⟨*base*⟩`,`⟨*altezza*⟩`)[`⟨*posizione*⟩`]{`⟨*testo*⟩`}` È una scatola delle dimensioni specificate mediante ⟨*base*⟩,⟨*altezza*⟩; il ⟨*testo*⟩ (che può essere un altro oggetto 'tipografico' qualsiasi, anche una fotografia introdotta mediante `\includegrapghics`) collocato in ⟨*posizione*⟩ dentro alla scatola mediante una o due delle lettere opzionali `c` (centre), `l` (left) e `r` (right), `t` (top), `b` (bottom). Il punto di rifermento è lo spigolo inferiore sinistro. Le dimensioni della scatola possono anche essere nulle e questa strana dichiarazione è forse la più usata in pratica; è molto comoda per collocare con precisione delle piccole legende nei disegni.

`\framebox(`⟨*base*⟩`,`⟨*altezza*⟩`)[`⟨*posizione*⟩`]{`⟨*testo*⟩`}` È una scatola con cornice, come quella vista per il testo, ma in questo caso se ne possono specificare le dimensioni sia orizzontali sia verticali; comodissima per gli schemi a blocchi. Il punto di rifermento è lo spigolo inferiore sinistro.

`\dashbox{`⟨*lung-trattino*⟩`}(`⟨*base*⟩`,`⟨*altezza*⟩`)[`⟨*posizione*⟩`]{`⟨*testo*⟩`}` È una scatola come `\framebox` ma con la cornice tratteggiata con trattini alternativamente bianchi e neri di lunghezza peri a ⟨*lungh-trattino*⟩; è opportuno che le lunghezze dei lati siano multipli dispari della lunghezza del trattino.

`\oval[`⟨*raggio*⟩`](`⟨*base*⟩`,`⟨*altezza*⟩`)[`⟨*parte*⟩`]` È un rettangolo con i vertici arrotondati mediante archi di cerchio di raggio pari a ⟨*raggio*⟩. Se ne può disegnare solo una ⟨*parte*⟩ mediante una o due delle lettere opzionali `t` (top), `b` (bottom), `l` (left), `r` (right). Specificando una lettera sola si disegna mezzo 'ovale'; specificando due lettere si disegna solo un quarto di 'ovale', purché le due lettere non siano inconsistenti (`tb` non avrebbe senso). In ogni caso il punto di riferimento è il centro dell''ovale', anche quando se ne disegna solo una parte.

`\arc[`⟨*angolo1*⟩`,`⟨*angolo2*⟩`]{`⟨*raggio*⟩`}` È un arco di circonferenza con il centro nel punto di riferimento; l'arco ha un raggio pari a ⟨*raggio*⟩ e comincia all'angolo ⟨*angolo1*⟩ e finisce all'angolo ⟨*angolo2*⟩.

`\arc*[`⟨*angolo1*⟩`,`⟨*angolo2*⟩`]{`⟨*raggio*⟩`}` È un settore circolare riempito del colore di default, generalmente nero; gli argomenti hanno lo stesso significato che hanno per `\arc`.

I comandi che non hanno bisogno di essere collocati nell'argomento di `\put` o `\multiput` sono i seguenti:

`\bezier{`⟨N⟩`}(`⟨Ax⟩`,`⟨Ay⟩`)(`⟨Bx⟩`,`⟨By⟩`)(`⟨Cx⟩`,`⟨Cy⟩`)` È una curva di Bézier di secondo grado; le coordinate x e y dei suoi tre punti A, B e C devono evidentemente essere specificate; nella versione del 'vecchio' LaTeX era *necessario* specificare il numero di punti ⟨N⟩ con cui disegnare la curva. Questo comando è conservato per compatibilità con il passato ma talvolta torna utile poter tratteggiare delle parabole a velocità crescente, e questo è il comando adatto.

`\qbezier[`⟨N⟩`](`⟨Ax⟩`,`⟨Ay⟩`)(`⟨Bx⟩`,`⟨By⟩`)(`⟨Cx⟩`,`⟨Cy⟩`)` È simile al comando precedente e si comporta nello stesso modo se viene specificato il valore opzionale del numero di punti con cui tracciare la curva. Altrimenti la curva viene tracciata facendo uso degli operatori PostScript oppure pdf a seconda di quale programma si stia usando per comporre il documento.

`\cbezier[`⟨N⟩`](`⟨Ax⟩`,`⟨Ay⟩`)(`⟨Bx⟩`,`⟨By⟩`)(`⟨Cx⟩`,`⟨Cy⟩`)(`⟨Dx⟩`,`⟨Dy⟩`)` È una curva di Bézier di terzo grado di cui vanno specificate obbligatoriamente le coordinate dei suoi quattro punti. Con $N = 0$ o non specificato vengono usati gli operatori interni del programma che si sta usando per comporre il documento.

`\Line(`⟨Ax⟩`,`⟨Ay⟩`)(`⟨Bx⟩`,`⟨By⟩`)` È il segmento che unisce i punti A e B.

`\polyline(`⟨Ax⟩`,`⟨Ay⟩`)(`⟨Bx⟩`,`⟨By⟩`)...(`⟨Zx⟩`,`⟨Zy⟩`)` È la spezzata che unisce i punti A, B,...,Z in numero qualunque, non limitato al numero delle 26 lettere dell'alfabeto.

`\polygon(`⟨Ax⟩`,`⟨Ay⟩`)(`⟨Bx⟩`,`⟨By⟩`)...(`⟨Zx⟩`,`⟨Zy⟩`)` È il contorno del poligono chiuso i cui vertici si susseguono nell'ordine A, B, ..., Z, A. Se si vuole

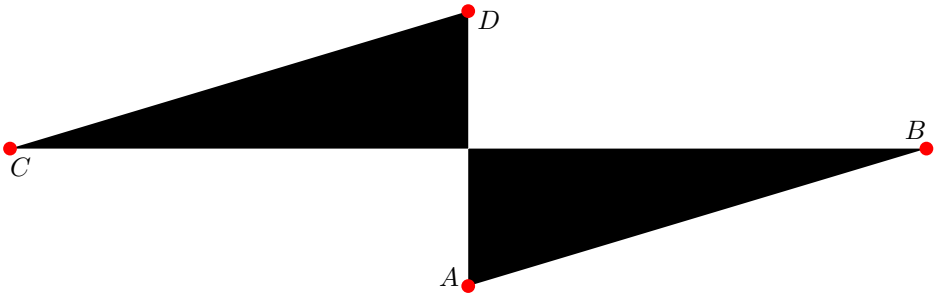

Figura 4.1: Poligono pieno con la spezzata del suo contorno intrecciata

vedere la cosa da un altro punto di vista, esso rappresenta la spezzata che si richiude sul punto di partenza.

`\polygon*`(⟨*Ax*⟩,⟨*Ay*⟩)(⟨*Bx*⟩,⟨*By*⟩)...(⟨*Zx*⟩,⟨*Zy*⟩) È il poligono pieno con gli stessi vertici descritti per `\polygon`; se la spezzata che descrive il contorno del poligono si intreccia, non è ben chiaro che cosa voglia dire che il poligono è 'pieno'. Il risultato si vede nella figura 4.1.

Vale la pena di commentare le curve di Bézier di secondo e di terzo grado. Quelle di secondo grado passano per i punti A e C con le tangenti specificate dai segmenti AB e BC, vedi figura 4.2; il punto B quindi funziona da punto guida, in quanto determina solo la direzione delle tangenti nei due punti di passaggio della curva che sono fissi. La curva è un arco di parabola che si svolge tutto all'interno del triangolo ABC.

Le curve di terzo grado sono curve più complesse che partono dal punto A e arrivano al punto D e i punti B e C funzionano da punti guida, cioè determinano la direzione delle tangenti nei due punti fissi estremi; la tangente alla curva nel punto A è data dal segmento AB, mentre la tangente nel punto D è data dal segmento CD, vedi figura 4.3. L'arco di curva si dovrebbe svolgere tutto all'interno del poligono individuato dalla sequenza dei vertici A, B, C e D; questi poligoni sono evidenziati con linee rosse nelle figure 4.2 e 4.3, ma siccome il contorno del poligono della figura 4.3 si intreccia, sembra che in corrispondenza dell'intreccio l'arco di curva esca un pochino, Quando il poligono non si intreccia, ma rappresenta un quadrilatero 'normale' l'arco di curva giace completamente dentro questo quadrilatero

Il codice per generare le figure 4.2 e4.3 è riportato nella figura 4.4.

Per essere svincolati da comandi precostituiti, si possono usare i comandi base con i quali si può virtualmente disegnare qualunque cosa; ovviamente i pacchetti dedicati PSTricks e pgf permettono di disegnare più agevolmente di quanto non posano fare i comandi di base. L'argomento è un po' troppo dettagliato per una guida iniziale come questa, quindi si rimanda alla documentazione.

Ma rimangono ancora alcuni importanti comandi che servono per caratterizzare le linee del disegno:

LaTeX in ambiente *picture* parte con lo spessore sottile per ogni linea retta o curva, per i bordi delle scatole, eccetera. Lo spessore di default è di 0,4 pt, molto sottile ma perfettamente visibile. Ricordiamo che un punto è circa 0,35 mm, quindi 0,4 pt equivalgono a circa 140 µm.

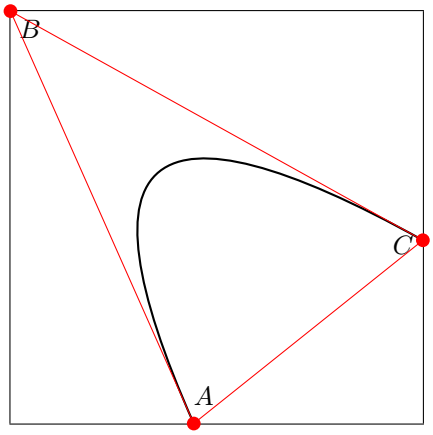

Figura 4.2: Curva di Bézier di secondo grado

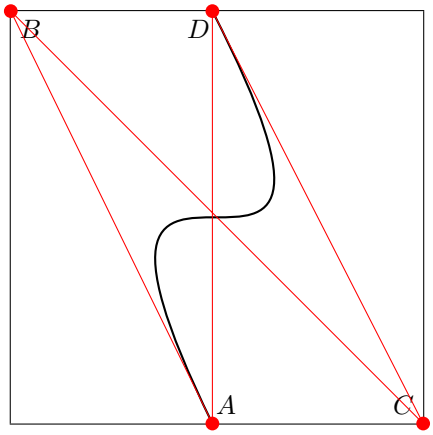

Figura 4.3: Curva di Bézier di terzo grado

`\thicklines` raddoppia lo spessore delle linee rette e curve.

`\thinlines` permette di tornare alle linee sottili.

`\linethickness{`⟨*spessore*⟩`}` permette di specificare lo ⟨*spessore*⟩ delle linee rette e curve; si noti che uno spessore di un punto tipografico, poco più di un terzo di millimetro, traccia già linee piuttosto scure, non parliamo di linee spesse un millimetro; infatti se lo spessore è 1 pt si ha: ——; invece se lo spessore è 1 mm si ha: ▬.

Nel disegno tecnico si usa procedere per spessori che aumentano secondo una progressione geometrica di ragione pari a $\sqrt{2} \approx 1{,}414$ Per un diagramma cartesiano, quindi, gli assi avranno uno spessore pari a 1 pt, il reticolato principale si userà uno spessore pari a 0,7 pt, e per il reticolato secondario uno spessore pari a 0,5 pt, ma per la curva tracciata sul diagramma si userà uno spessore pari a 1,4 pt, forse anche uno spessore di 2 pt. È quindi importante poter specificare lo spessore di tutte le linee che si devono tracciare.

Per una più completa documentazione del pacchetto `pict2e` si rinvia alla documentazione sul disco fisso nel file `.../doc/latex/pict2e/pict2e.pdf`.

Ora vediamo di fare una cosa come quelle che possono essere necessarie in un lavoro tecnico scientifico: tracciamo un diagramma cartesiano; per esprimere meglio le unità di misura scegliamo `\unitlength=1cm`, così le coordinate decimali hanno più senso. Tracciamo il diagramma del moto uniformemente accelerato con partenza da fermo corrispondente all'espressione

$$s = \frac{1}{2}at^2$$

con s espresso in metri, t in secondi e con l'accelerazione $a = 2\,\mathrm{m/s^2}$. Se il diagramma viene tracciato per l'intervallo di tempo da 0 s a 10 s, l'equazione ci dice che verrà percorso uno spazio di 100 m. La scala per le ascisse può avere un rapporto grafico di 1 cm/s mentre per le ordinate potremo scegliere un rapporto grafico di $1\,\mathrm{cm} \triangleq 10\,\mathrm{m}$.

Bisognerà dunque predisporre un diagramma su un reticolo di linee coordinate dalle quali si possano leggere i valori; gli assi andranno graduati; gli spessori delle

```
\begin{figure}\unitlength.01\textwidth
\begin{minipage}[t]{.45\textwidth} %%%%% prima figurina
\begin{picture}(45,45)
\put(0,0){\framebox(45,45){}}
\thicklines
\qbezier(20,0)(0,45)(45,20)
\put(20,2){\makebox(0,0)[bl]{$A$}}
\put(1,44){\makebox(0,0)[tl]{$B$}}
\put(44,20.5){\makebox(0,0)[tr]{$C$}}
\thinlines
\color{red}
\put(20,0){\circle*{1.5}}
\put(0,45){\circle*{1.5}}
\put(45,20){\circle*{1.5}}
\polygon(20,0)(0,45)(45,20)
\end{picture}
\caption{Curva di Bézier di secondo grado}
\label{fig:bezier2}
\end{minipage}
\hfill
\begin{minipage}[t]{.45\textwidth}%%%%%%%% seconda figurina
\begin{picture}(45,45)
\put(0,0){\framebox(45,45){}}
\thicklines
\cbezier(22,0)(0,45)(45,0)(22,45)
\put(23.5,1){\makebox(0,0)[b]{$A$}}
\put(1,44){\makebox(0,0)[tl]{$B$}}
\put(44,1){\makebox(0,0)[br]{$C$}}
\put(20.5,44){\makebox(0,0)[t]{$D$}}
\thinlines\color{red}
\put(22,0){\circle*{1.5}}
\put(0,45){\circle*{1.5}}
\put(45,0){\circle*{1.5}}
\put(22,45){\circle*{1.5}}
\polygon(22,0)(0,45)(45,0)(22,45)
\end{picture}
\caption{Curva di Bézier di terzo grado}
\label{fig:bezier3}
\end{minipage}
\end{figure}
```

Figura 4.4: Codice per generare le figure 4.2 e4.3

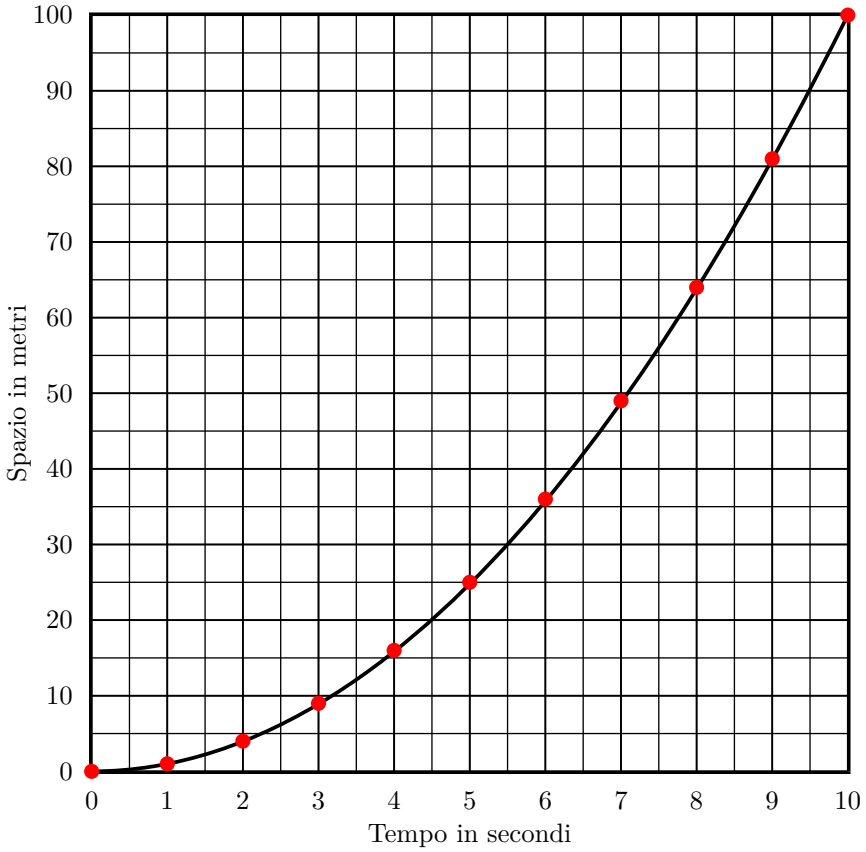

Figura 4.5: Moto uniformemente accelerato. I pallini rossi indicano i punti della curva determinati numericamente.

linee coordinate e della curva dovranno essere conformi alle regole del disegno tecnico descritte sopra. Otterremo dunque il diagramma della figura 4.5. Il codice è riportato nella figura 4.6.

Si noti che se si guarda la figura 4.5 sullo schermo del PC, che ha una risoluzione abbastanza bassa, non si percepiscono i diversi spessori delle linee del reticolo; ma se si usa lo strumento per ingrandire la schermata, si può vedere benissimo il risultato dei diversi spessori. A stampa, anche solo a 300 punti al pollice, questi difetti in generale non appaiono.

Si osservi anche l'uso del comando `\rotatebox{⟨angolo⟩}{⟨oggetto⟩}` per ruotare di 90° la legenda dell'asse verticale; la scelta delle dimensioni nulle per la scatola che contiene la legenda permette di avere un punto preciso attorno al quale eseguire la rotazione. Il comando `\rotatebox` è reso disponibile dal pacchetto graphicx che si deve comunque caricare per importare le fotografie e altri tipi di immagini.

```
\begin{figure}\unitlength=1cm\centering
\begin{picture}(11,11)(-1,-1)
\linethickness{0.35pt}
\multiput(0,0)(0.5,0){21}{\line(0,1){10}}
\multiput(0,0)(0,0.5){21}{\line(1,0){10}}
\linethickness{0.5pt}
\multiput(0,0)(1,0){11}{\line(0,1){10}}
\multiput(0,0)(0,1){11}{\line(1,0){10}}
\linethickness{0.7pt}
\put(0,0){\framebox(10,10){}}
\put(0,-0.5){\makebox(0,0)[b]{0}}
\put(1,-0.5){\makebox(0,0)[b]{1}}
\put(2,-0.5){\makebox(0,0)[b]{2}}
\put(3,-0.5){\makebox(0,0)[b]{3}}
\put(4,-0.5){\makebox(0,0)[b]{4}}
\put(5,-0.5){\makebox(0,0)[b]{5}}
\put(6,-0.5){\makebox(0,0)[b]{6}}
\put(7,-0.5){\makebox(0,0)[b]{7}}
\put(8,-0.5){\makebox(0,0)[b]{8}}
\put(9,-0.5){\makebox(0,0)[b]{9}}
\put(10,-0.5){\makebox(0,0)[b]{10}}
\put(5,-1){\makebox(0,0)[b]{Tempo in secondi}}
\put(-0.25,0){\makebox(0,0)[r]{0}}
\put(-0.25,1){\makebox(0,0)[r]{10}}
\put(-0.25,2){\makebox(0,0)[r]{20}}
\put(-0.25,3){\makebox(0,0)[r]{30}}
\put(-0.25,4){\makebox(0,0)[r]{40}}
\put(-0.25,5){\makebox(0,0)[r]{50}}
\put(-0.25,6){\makebox(0,0)[r]{60}}
\put(-0.25,7){\makebox(0,0)[r]{70}}
\put(-0.25,8){\makebox(0,0)[r]{80}}
\put(-0.25,9){\makebox(0,0)[r]{90}}
\put(-0.25,10){\makebox(0,0)[r]{100}}
\put(-0.8,5){\rotatebox{90}{\makebox(0,0)[b]{Spazio in metri}}}
\linethickness{1.4pt}
\cbezier(0,0)(4.5,0)(8,6)(10,10)
\color{red}
\put(0,0){\circle*{0.2}}
\put(1,0.1){\circle*{0.2}}
\put(2,0.4){\circle*{0.2}}
\put(3,0.9){\circle*{0.2}}
\put(4,1.6){\circle*{0.2}}
\put(5,2.5){\circle*{0.2}}
\put(6,3.6){\circle*{0.2}}
\put(7,4.9){\circle*{0.2}}
\put(8,6.4){\circle*{0.2}}
\put(9,8.1){\circle*{0.2}}
\put(10,10){\circle*{0.2}}
\end{picture}
\caption{Moto uniformemente accelerato}\label{fig:motoaccelerato}
\end{figure}
```

Figura 4.6: Codice per il disegno della figura 4.5

Capitolo 5

Le presentazioni

5.1 Che cosa è una presentazione

Il mark-up di LaTeX, specialmente attraverso il programma di composizione **pdflatex**, mediante la classe *beamer*[1] permette di creare delle magnifiche presentazioni da proiettare direttamente dal PC attraverso un proiettore per ambienti più o meno grandi.

Naturalmente molti conoscono e/o hanno usato un noto programma commerciale per la produzione di presentazioni al calcolatore. Esso è un prodotto molto valido, ma come tutti i prodotti della stessa suite fa una certa fatica a comporre il testo matematico.

Ecco perché una presentazione tecnico scientifica ha assoluto bisogno di uno strumento legato al sistema TeX, che permetta di produrre *slide* con contenuto matematico in modo perfetto. Siccome il file che contiene la presentazione sarà preferibilmente in formato pdf, così che il presentatore si possa portare dietro alla conferenza solo una chiavetta USB, senza correre il rischio che sul PC del "congresso" manchi il software gratuito della Adobe, l'onnipresente **Adobe Reader**, quando invece sullo stesso PC del "congresso" potrebbe mancare la famosa suite di programmi commerciali, o potrebbe essere presente una versione datata non adatta alle prestazioni del programma che avete usato. Notate che questo vale anche per le presentazioni predisposte con la suite *open source* OpenOffice.org, in particolare con il suo modulo **Impress**; invece il linguaggio PDF interpretato e trasformato in immagini da **Adobe Reader** dovrebbe essere sempre in grado di proiettare la presentazione in modalità a pieno schermo.

5.2 La classe *beamer*

La classe *beamer* si usa come le classi `article` o *book*; semplicemente è dotata di una collezione di moduli con varie prestazioni per gestire lo stile delle diapositive, i colori, il tipo di hyperlink interni per navigare fra le diapositive in modo non sequenziale, per gestire le dissolvenze/transizioni, per gestire gli *overlay* (materiale che appare in schermate successive, ma che compone una sola diapositiva), eccetera.

[1] In inglese *beamer* significa *proiettore*.

Il file sorgente della presentazione è sempre separato in preambolo e corpo della presentazione. Il preambolo contiene come al solito tutto ciò che può servire per confezionare il contenuto, cioè il corpo della presentazione. Siccome la classe è stata predisposta apposta per formattare la presentazione in pdf, esistono anche dei comandi che eseguono (all'insaputa del compositore) dei comandi del linguaggio PDF che consentono non solo la navigazione, ma anche la gestione più efficace delle immagini, specialmente quelle che devono venire usate più volte, per esempio, il logo della istituzione di appartenenza dell'oratore, che probabilmente apparirà in un angolo di ogni diapositiva.

Con *beamer* bisogna distinguere fra *frame* e *slide*; queste ultime sono le singole pagine pdf che contengono un'intera diapositiva o parte di essa; *frame* è la schermata che può consistere di una sola diapositiva o dalla sovrapposizione (*overlay*) di diverse *slide*. I comandi per creare un *frame*, composto di una o più *slide*, sono racchiusi dentro all'ambiente *frame* il cui argomento obbligatorio contiene il suo titolo.

5.3 Esempio di presentazione

Prendendo spunto da qualche pagina del testo [1], il file sorgente potrebbe apparire così:

```
\documentclass{beamer}
%                                    Preambolo
\usetheme{AnnArbor}
\useoutertheme[right]{sidebar}
\setbeamercolor{alerted text}{fg=red!90!black}
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}
\usepackage{pgf}
\usepackage{pict2e,curve2e}
\usepackage[color]{guit}
\usepackage[italian]{babel}
%
\usefonttheme{professionalfonts}
\usepackage{lxfonts}

\beamertemplatetransparentcovereddynamic

\title{I numeri complessi}
\subtitle{Conferenza internazionale
del~Gruppo~Italiano~degli~utenti~di~\TeX}
\author{Mario Rossi}
\institute{\GuIT}

\date{Pisa, 20--22 ottobre 2015}


\pgfdeclareimage[width=\textwidth]{CifreIndiane}%
                                {CifreIndianeVIIsecolo}
\pgfdeclareimage[width=15.5mm]{guitlogo}{GuITlogo}
```

```
\logo{\pgfuseimage{guitlogo}}
% Fine del preambolo

\begin{document}% Inizio della presentazione


\begin{frame}% Primo frame, prima slide
\titlepage
\end{frame}

\begin{frame}% Secondo frame, seconda slide
\frametitle{Indice}
\tableofcontents
\end{frame}

\section{Introduzione storica}

\begin{frame}% Terzo frame, terza slide
\frametitle{Una breve storia dei numeri}
All'inizio del \textsc{vii} secolo gli Indiani inventarono
la notazione posizionale e le "<nove"> cifre dall'1 al 9;
lo zero veniva detto a voce ma non aveva ancora un suo segno.

\begin{center}
\pgfuseimage{CifreIndiane}
\end{center}
\end{frame}

\section{Nascita dei numeri complessi}


\begin{frame}% Quarto frame,  diviso in quattro slide
\frametitle{I numeri complessi nascono nel '500}
\begin{enumerate}
\item<1-> Nel XVI secolo Tartaglia e Cardano introducono
la radice quadrata di $-1$
\item<2-> Il nome di \emph{unità immaginaria} viene creato
da Réné Descartes nel 1637
\item<3-> Gauss nel 1799 contribuisce con i suoi scritti
a diffondere i numeri complessi
\item<4> Hamilton nel 1833 pubblica la teoria dei numeri
complessi
\end{enumerate}
\end{frame}


\section{I numeri complessi in \texttt{pict2e}}

\begin{frame}% Quinto frame, ottava slide
```

```
\frametitle{I numeri complessi come operatori geometrici}
\begin{center}\unitlength=1mm
   \begin{picture}(60,40)
   \put(0,0){\vector(1,0){60}}\put(60,1){\makebox(0,0)[br]{$x$}}
   \put(0,0){\vector(0,1){45}}\put(1,45){\makebox(0,0)[lt]{$y$}}
   \thicklines
   \put(0,0){\textcolor{red}{\vector(3.4641,2){40}}}
   \thinlines
   \multiput(40,-.5)(0,2){12}{\line(0,1){1}}
   \multiput(-.5,23.094)(2,0){20}{\line(1,0){1}}
   \put(41,1){\makebox(0,0)[bl]{$a$}}
   \put(1,24.1){\makebox(0,0)[bl]{$b$}}
   \VectorArc(0,0)(20,0){30}
   \put(20,4){\makebox(0,0)[bl]{$\varphi$}}
   \put(20,12.6){\rotatebox{30}{\makebox(0,0)[b]{$m$}}}
   \put(0,0){\thicklines\textcolor{blue}{\vector(1,0){10}}}
   \put(5,-1){\textcolor{blue}{\makebox(0,0)[t]{vettore
              unitario}}}
   \end{picture}
\end{center}
Visto come operatore geometrico, il numero complesso
$m\mathsf{e}^{\mathsf{i}\varphi}=a+\mathsf{i}b$ agisce su
un vettore; in questa figura agisce sul
\textcolor{blue}{vettore unitario blu}; lo scala tramite
il fattore \alert{$m$} e lo ruota dell'angolo \alert{$\varphi$}
producendo il \alert{vettore rosso}.
\end{frame}

\end{document}
```

Quei comandi servono per produrre le otto *slide* dei cinque *frame* presentati in piccolo nella figura 5.1.

Certamente il listato del programma potrebbe apparire ostico; qualche commento serve per rendere la cosa più comprensibile.

Dopo la dichiarazione della classe appaiono alcune impostazioni; quelle relative all'*input encoding*, al *font encoding* e alla lingua ci sono ormai familiari, le altre sono impostazioni per lo stille delle diapositive; per scegliere fra i vari stili la documentazione di beamer è ricchissima e vi si rimanda.

Si è già parlato anche dei pacchetti pgf e pict2e; curve2e è un'estensione del precedente. Il pacchetto guit serve per il logo della associazione Gruppo Utenti Italiani di TEX e LATEX.

Interessanti sono le due righe:

```
\usefonttheme{professionalfonts}
\usepackage{lxfonts}
```

dove si dichiara di voler usare dei *professional fonts*; dei font che non sono solo testuali, ma comprendono anche le varie parti per comporre la matematica e i simboli testuali accessori; il pacchetto lxfonts serve per invocare tutti i file necessari per usare questi font (incidentalmente prodotti da un membro del

Figura 5.1: Otto slide per una presentazione di cinque frame

Gruppo degli Utilizzatori Italiani di TEX e LATEX) particolarmente adatti alla composizioni della matematica nelle diapositive.

Le righe:

```
\pgfdeclareimage[width=\textwidth]{CifreIndiane}%
                                  {CifreIndianeVIIsecolo}
\pgfdeclareimage[width=15.5mm]{guitlogo}{GuITlogo}
\logo{\pgfuseimage{guitlogo}}
```

servono per impostare una volta per tutte due figure specificando nell'ultimo argomento il nome del file grafico senza estensione (che in realtà sarebbe una di quelle ammissibili da **pdflatex**). Nel copro della presentazione esse verrano richiamate solo mediante il loro "nome" attribuito loro con il secondo argomento; l'argomento facoltativo serve per impostare la dimensione della figura. In questo modo l'inserimento delle figure dentro i *frame* risulta facilitato; in particolare per il logo del GuIT, il comando `\logo` permette di usarlo in ogni *slide* così che appaia in tutti i *frame*, benché il file sia stato memorizzato una volta sola e venga richiamato solamente attraverso gli hyperlink interni del formato pdf.

Il resto, il corpo della presentazione, non richiede particolari commenti. È invece interessante osservare come il quarto *frame* sia formato da quattro *slide*, separatamente individuabili nella figura 5.1, ma descritte da un solo ambiente *frame* nel file sorgente; i commenti permettono di seguire meglio la separazione prodotta dai vari comandi; ma la "magia" è semplicemente ottenuta dai vari comandi `\item` della lista puntata che contengono un argomento facoltativo il quale serve per specificare quando ogni *slide* deve essere mostrata: `\item<1->` dice che la prima *slide* deve essere presentata nel *frame* dalla prima schermata in poi; `\item<2->` dalla seconda schermata in poi, eccetera, fino a `\item<4>` che si riferisce alla sola quarta (e ultima) schermata.

Non sono presenti transizioni; di queste non bisogna abusare, come si fa comodamente con altri programmi di creazione di presentazioni; le transizioni servono ad attirare l'attenzione dello spettatore sulla transizione stessa e lo distraggono dal contenuto delle diapositive. al più vanno usate quando nella presentazione si passa ad un argomento totalmente nuovo e distinto/scorrelato dal precedente.

Bibliografia

[1] Claudio Beccari, (A cura di). *Introduzione all'arte della composizione tipografica con LaTeX*. Gruppo Utilizzatori Italiani di TeX e LaTeX, 2010. http://www.guit.sssup.it/downloads/GuidaGuIT.pdf.

[2] Lorenzo Pantieri. *LaTeX per l'impaziente – Un'introduzione all'Arte di scrivere con LaTeX*. Gruppo Utilizzatori Italiani di TeX e LaTeX, settembre 2009. http://www.lorenzopantieri.net/LaTeX_files/LaTeXimpaziente.pdf.

[3] Lorenzo Pantieri e Tommaso Gordini. *L'arte di scrivere con LaTeX*, 2010. http://www.lorenzopantieri.net/LaTeX_files/ArteLaTeX.pdf.

[4] Tobias Ötlicher. *Una (mica tanto) breve introduzione a LaTeX 2ε Ovvero LaTeX 2ε in 93 minuti*. TeX Users Group. In $TEXMF/guides/lshort/itlshort.pdf. Traduzione a cura di G. Agostini, G. Bilotta, F. Casadei Della Chiesa, O. de Bari, G. Delre, L. Ferrante, T. Pecorella, M. Rigido, R. Zanasi.