Claudio Beccari

Comporre tabelle e matrici

Come usare LaTeX per comporre tabelle e matrici ben fatte

Versione 1.1.02 del 2021-09-22

Associati anche tu al GUIT

Fai click per associarti

L'associazione per la diffusione di TEX in Italia, riconosciuta ufficialmente in ambito internazionale, si sostiene *unicamente* con le quote sociali.
Se anche tu trovi che questa guida tematica gratuita ti sia stata utile, il mezzo principale per ringraziare gli autori è diventare socio.

Divenendo soci si ricevono gratuitamente:

- l'abbonamento alla rivista ArsTEXnica;
- il DVD TEX Collection;
- un eventuale oggetto legato alle attività del GUIT.

L'adesione al GUIT prevede un quota associativa compresa tra 12,00 € e 70,00 € a seconda della tipologia di adesione prescelta e ha validità per l'anno solare in corso.

Comporre tabelle e matrici
Copyright © 2021, Claudio Beccari

Questa documentazione è soggetta alla licenza LPPL (LATEX Project Public Licence), versione 1.3 o successive; il testo della licenza è sempre contenuto in qualunque distribuzione del sistema TEX e nel sito http://www.latex-project.org/lppl.txt.

Questo documento è curato da Claudio Beccari.

Presentazione

Le tabelle sono sempre state un problema per i tipografi, anche quando componevano a mano con i caratteri mobili.

Con il linguaggio LaTeX e con i vari pacchetti disponibili tramite un'installazione completa e aggiornata del sistema TeX, i problemi dovrebbero essere minori, ma questo non toglie che le tabelle siano sempre delicate da comporre.

Inoltre, l'uso diffuso dei noti word processor ci ha abituati a vedere tabelle composte male, con ogni casella contornata dal suo bordo reso ben visibile da filetti verticali e orizzontali. In tipografia si usano i filetti con molta parsimonia e con uno scopo ben preciso, non in modo indiscriminato come avviene con i word processor.[1]

Ci sono vari tipi di tabelle: numeriche, testuali, miste, con grandezze misurabili, con i valori numerici incolonnati sulla base della loro cifra delle unità, con celle testuali che si svolgono su più colonne, con celle testuali che impegnano diverse righe di testo, con celle che si sviluppano in verticale e impegnano diverse righe di celle; tabelle che devono restare oggetti monolitici oppure che si svolgono su più pagine mantenendo fissi gli incolonnamenti. Insomma, non c'è che l'imbarazzo della scelta.

L'autore di un documento deve quindi scegliere il tipo di tabella che va bene per i suoi dati e deve progettarla in anticipo avendo sempre a mente il lettore; questi non deve essere confuso con tabelle contenenti materiale eterogeneo, deve poter contare su intestazioni delle colonne concise ma esplicative, non deve cavarsi gli occhi per leggere materiale troppo fitto e scritto con caratteri troppo piccoli. Deve poter inclinare il capo sempre dallo stesso lato quando legge tabelle ruotate sempre nello stesso verso indipendentemente dalla pagina pari o dispari in cui si trovano.

Insomma, la composizione di tabelle richiede molte analisi preliminari prima che l'autore cominci a comporle.

[1] I word processor sono più facili da usare di ogni programma del sistema TeX, ma hanno prestazioni decisamente limitate. Si può fare un lavoro accettabile anche con i word processor, ma bisogna lavorare molto di più, senza limitarsi a qualche click di mouse.

Il sistema TEX offre diversi pacchetti che permettono di comporre tabelle. Non verranno descritti tutti, ma cercherò di segnalare quelli più usati per i diversi compiti che l'autore deve affrontare. In ogni caso segnalerò solo i pacchetti che ho usato più o meno a lungo e di cui ho abbastanza esperienza per parlarne. Indicherò anche alcuni trucchi che ho usato e che ho sfruttato per comporre questa guida tematica.

Mi rendo conto di avere scritto più di 100 pagine ma, nonostante ciò, di essere restato in superficie. Perché solo sulle tabelle si potrebbero scrivere tomi pesantissimi, mentre una guida tematica dovrebbe rimanere entro limiti ristretti, puntando sulla profondità e non sull'estensione del materiale di cui tratta.

In effetti, spero che il lettore si renda conto di quanto lavoro ci sia dietro il software del sistema TEX anche limitandosi alla composizione delle tabelle. Spero che il lettore si renda conto che tutti i pacchetti che sono stati sviluppati attorno a questo argomento hanno dei pregi e dei difetti, oltre che dei limiti; il sistema TEX non è onnipotente, anche se è in grado di gestire quasi tutto ciò che può essere stampato. Il suo limite maggiore, che è anche il suo pregio, è che si occupa di composizione tipografica, un'arte finalizzata a comunicare attraverso il segno scritto in maniera tale da agevolare il lettore nella comprensione del messaggio che l'autore desidera trasmettere.

Il sistema TEX non è un word processor, col quale chiunque può giocare per essere orgoglioso di quel che riesce a scrivere; sono abbastanza vecchio per aver imparato da ragazzo ad usare una macchina da scrivere, con la quale ottenevo pagine scritte in modo leggibile, ma certamente non belle; inoltre usavo una macchina che scriveva solo in maiuscoletto...

Quando ho avuto per le mani il mio primo calcolatore e il programma $\text{Word}_{\text{Star}}$, mi pareva di toccare il cielo con un dito; mi sono servito di quello per scrivere il mio primo libro; per fortuna l'editore lo ha ricomposto tipograficamente e mi sono reso conto che il mio $\text{Word}_{\text{Star}}$ era solo una macchina da scrivere un filino più 'intelligente'. Poi ho incontrato LATEX e ho capito che la tipografia è decisamente un'altra cosa; all'inizio non riuscivo a staccarmi dalle 'cattive abitudini' dello scrivere a macchina. Mi ci sono voluti anni per affinare il gusto e non dico di essere arrivato alla fine del percorso, perché continuo a essere un dilettante, ma ho imparato molto. Anche la composizione delle tabelle mi ha insegnato a migliorarmi: con questa guida tematica cerco di trasmettere quello che ho imparato.

Il piano di questa guida tematica è il seguente.

1. Il primo capitolo inizia la guida facendo chiarezza fra gli ambienti

table e *tabular*; il novizio fa fatica a distinguerli, poi con la pratica la confusione cessa. Tuttavia la molteplicità delle tabelle/tabulazioni/incolonnamenti che si possono fare con LaTeX diventano chiari, ma ogni tipo richiede una attenta progettazione.

2. Infatti il secondo capitolo parla della progettazione delle tabelle indicando anche quali problemi si incontrano se la progettazione non è adatta. Indica anche i primi metodi per ovviare a tali problemi, ma i metodi più complessi verrano trattati nei capitoli successivi.
3. Il terzo capitolo parla delle tabelle create con gli ambienti definiti nel kernel di LaTeX: *tabular* e *tabular**. Vengono mostrati numerosi esempi cercando di evidenziare le differenze.
4. Nel quarto capitolo si parla degli ambienti *tabularx* e *widetabular* che affrontano il problema della costruzione di tabelle di larghezza prefissata; gli approcci dei due ambienti sono complementari e hanno campi di applicazione diversi; di nuovo gli esempi permettono di rendersi conto delle differenze.
5. Nel quinto capitolo si parla degli incolonnamenti in ambito matematico, cioè delle matrici, di sistemi di equazioni e di costrutti affini.
6. Nel sesto capitolo si parla di come costruire tabelle che sporgano nel margine; è un approccio che consente di ottenere tabelle più larghe della giustezza del testo fino ad invadere lo spazio nel margine esterno dedicato ale note marginali; si accenna anche a come comporre piccole tabelle parzialmente sporgenti nel margine e parzialmente contornate dal testo.
7. Nel settimo capitolo si descrive come ruotare le tabelle larghe, ma non troppo alte; la rotazione in senso antiorario nella pagina può risolvere diversi problemi.
8. Nell'ottavo capitolo si descrivono le tabelle che si sviluppano su più pagine; evidentemente non possono venire composte con i metodi descritti nei capitoli precedenti, perché la loro composizione richiede algoritmi completamente diversi implementati dagli ambienti *longtable* e *supertabular*.
9. Nel nono capitolo si parla del poco noto ambiente *tabu* che da solo dovrebbe poter fare ciò che si fa con gli ambienti descritti nei capitoli precedenti e fa anche qualcosa in più. Alcuni esempi ne mostrano le funzionalità.
10. Infine nel decimo capitolo si parla degli incolonnamenti costruiti con l'ambiente *tabbing* che si rifà ai metodi usati con le macchine da

scrivere dattilografiche. Si tratta di un ambiente poco noto e poco usato ai nostri giorni, ma era quello che agevolava la composizione di incolonnamenti agli inizi dell'esistenza di LaTeX, quando ancora, specialmente negli uffici, il personale preferiva usare qualcosa simile a quanto era abituato fare prima di essere "esposto" alle novità dell'elaborazione computerizzata.

Questa guida non contiene nessuna bibliografia; l'unico testo che viene menzionato è la guida integrale di Leslie Lamport, *LaTeX: A document preparation system – User guide and reference manual*, seconda edizione del 1994, pubblicata dalla Addison Wesley. La parte *reference manual*, l'appendice C di quel testo, è stata tradotta in italiano con abbondanza di commenti nella guida tematica *Il LaTeX Handbook Reference Manual commentato*; è scaricabile dalla sezione Documentazione del forum del GuIT: www.guitex.org. Va notato, però, che malgrado l'abbondanza di commenti, la parte dedicata alle tabelle e alle tabulazioni è limitata, anche se quella guida tematica ne contiene molte; alcune si svolgono su più pagine, talvolta ricorrendo ad artifici un po' acrobatici per inserirvi elementi che sarebbe stato difficile descrivere nella presente guida; erano e rimangono artifici specifici di quelle tabelle, probabilmente inutili in qualsiasi altra circostanza.

Tuttavia il lettore ricordi sempre che ogni pacchetto che desidera usare è completamente documentato mediante file che fanno già parte dell'installazione del suo sistema TeX. Non c'è bisogno di andare a cercare in rete per trovare delucidazioni o istruzioni. Chiunque usi LaTeX ha già tutto quello che gli serve a bordo del suo calcolatore; basta che dia il comando `texdoc` ⟨*nome del pacchetto*⟩ attraverso la finestra di un terminale, console, xterm, prompt dei comandi, comunque si chiami con il sistema operativo della macchina in uso, e la documentazione originale del pacchetto appare immediatamente. Va da sé che, se si dispone di una installazione parziale del sistema TeX, il comando `texdoc` potrebbe non trovare la documentazione cercata. Chi si affida ai servizi di compilazione on line in generale dispone di funzionalità simili, ma non è detto che quei servizi usino software completo e aggiornato.

Spero che questa "piccola" guida tematica possa essere d'aiuto ai numerosi utenti di LaTeX italiani.

22 giugno 2019

Claudio Beccari
`claudio dot beccari at gmail com`

Indice

Elenco delle tabelle

L'ambiente *table* e le tabelle 1

Per il neofita che comincia a usare LaTeX talvolta è difficile separare l'ambiente *table* dal concetto di creare una tabella. Sgombriamo quindi il campo da quest'ambiguità.

L'ambiente *table* serve solo per rendere flottante il contenuto dell'ambiente mettendoci una didascalia con il comando `\caption` la cui prima parte comincia con la parola Tabella. Il contenuto dell'ambiente è a totale descrizione dell'utente: potrebbe essere per davvero una tabella, oppure l'immagine di una tabella, o una tabella affiancata a una figura, insomma qualunque cosa che abbia senso di essere etichettata con una didascalia che inizia con Tabella.

Gli oggetti flottanti sono molto particolari; essi non inseriscono subito il loro contenuto nel file composto, ma lo trattengono in memoria fino a quando lil programma di composizione trova il posto giusto per inserirli nel file di uscito; i criteri con cui LaTeX considera "giusto" il posto dove inserire il suo contenuto sono molto complessi e non vengono descritti in questa documentazione; se l'autore vuole modificare questi criteri, lo può fare, ma deve sapere esattamente che cosa sta facendo e le conseguenze delle sue modifiche. Può trovare spiegazioni sommarie sia nella guida tematica *Il LaTeX Reference Manual commentato* (sempre scaricabile dalla sezione Documentazione del sito del GuIT) sia dalla guida generale *Introduzione all'arte della composizione tipografica con LaTeX* (anche questa scaricabile dallo stesso sito).

Va specificato subito che nessun ambiente flottante accetta né le note in calce né le note marginali. Tuttavia nelle tabelle talvolta sono necessarie alcune note specifiche, quindi bisogna trovare il modo per aggirare la limitazione, in particolare, dell'ambiente *table*. Se ne parlerà più avanti; il lettore, però, si appunti subito questo problema, affinché non resti spiazzato

se le sue normali note spariscono completamente, tranne solo il loro numero di richiamo.

Se si vuole comporre una tabella vera e propria, da far flottare tramite l'ambiente *table*, o da tenere fuori testo nel punto esatto dove si vuole scomporla, bisogna ricorrere ad ambienti adeguati o a opportuni comandi.

Un avviso: qualunque cosa che contenga colonne di dati testuali, matematici o numerici potrebbe essere chiamata "tabella"; ricordando le vecchie macchine da scrivere, tutte le cose potevano essere incolonnate mediante il tabulatore. Questo in qualche modo è sopravvissuto nei calcolatori moderni, tanto che la tastiera sulla sinistra ha il tasto `Tab`.

Ma il sistema TEX crea gli incolonnamenti sia con l'ambiente *tabbing*, che rispecchia le funzionalità delle macchine da scrivere, sia con i vari ambienti specifici per comporre tabelle di diversi tipi. L'ambiente più noto è l'ambiente *tabular* con o senza asterisco, ma ce ne sono diversi altri con funzionalità particolari. Qui c'è una brevissima descrizione degli ambienti che verranno descritti in questa guida.

tabular Compone una tabella determinando automaticamente la larghezza di ogni colonna definendo gli allineamenti del contenuto delle celle di ciascuna colonna. Le celle possono contenere qualsiasi cosa, anche della matematica, ma è preferibile usare *tabular* per contenuti testuali.

array Funziona come *tabular* ma deve essere usato in modo matematico e le celle contengono formule o altri oggetti matematici; serve specialmente per comporre matrici e formulari; ma per questo scopo gli ambienti definiti dal pacchetto `amsmath` sono decisamente più efficaci.

*tabular** È un'estensione dell'ambiente *tabular* che consente di allargare la tabella perché abbia una larghezza specificata. I descrittori degli allineamenti dei contenuti delle celle sono identici a quelli dell'ambiente *tabular*.

tabularx Costruisce tabelle di larghezza specificata ricorrendo a un nuovo tipo di descrittore di colonna, adatto essenzialmente per il testo; questo descrittore è capace di definire una colonna di cui calcola la larghezza in modo automatico per raggiungere la larghezza complessiva specificata per la tabella.

widetable Costruisce tabelle di larghezza specificata ricorrendo all'ampliamento dello spazio intercolonna. I descrittori di colonna continuano

ad essere quelli di *tabular* ma l'allargamento viene ottenuto in modo complementare a quello usato da *tabularx*.

longtable Costruisce una tabella da non flottare, capace di eseguire automaticamente i salti di pagina necessari per tabelle che si sviluppano su più pagine.

supertabular Similmente a *longtable*, produce tabelle che si sviluppano su più pagine, ma usa una sintassi diversa e un diverso algoritmo per spezzare la tabella a fine pagina.

tabu È un unico ambiente che accetta un certo numero di opzioni per gestire il contenuto di una tabella in modo da comporla come uno qualsiasi degli ambienti precedenti, ma fornendo anche ulteriori funzionalità. È poco usato ma forse potrebbe essere usato più spesso se gli autori lo conoscessero meglio. Si precisa che questo ambiente ha moltissime funzionalità, per cui anche la sua documentazione è relativamente lunga; in effetti l'uso di questo ambiente richiede che l'autore si faccia una certa esperienza.

Naturalmente, per alcuni di questi ambienti bisogna caricare appositi pacchetti che li definiscono; gli unici che fanno parte nativamente del linguaggio LaTeX sono *tabular*, *tabular** e *array*.[1]

[1] Non si confonda il nome del pacchetto `array` con quello dell'ambiente *array*.

Progettazione di una tabella 2

Come anticipato, prima di comporre una tabella bisogna progettarla per poter sceglierne la composizione e lo stile migliori.

Non bisogna dimenticare che la tabella dovrebbe stare nella giustezza del testo che si sta componendo; che sia una tabella resa flottante, o una tabella che si estende su più pagine, bisogna conoscere esattamente la giustezza della gabbia che contiene il testo.

Le dimensioni della gabbia sono essenziali; in un foglio A4 di 210mm per 297mm la gabbia, esclusi testatina e piedino, potrebbe essere di 160mm per 230mm; le dimensioni precise sono date dalla classe o dalle impostazioni scelte dall'autore mediante le funzionalità della classe o di pacchetti come `geometry`. In questa documentazione, composta in formato B5, di 176mm per 250mm, la gabbia è un rettangolo di `123.71451 mm` per `175.94179 mm`.[1]

Quindi, in questo documento ogni tabella non può essere più larga di circa 124mm. Se la tabella fosse più larga, ma non più di circa 176mm, né più alta di circa 110mm, si potrebbe pensare di ruotarla di 90° in senso antiorario, indipendentemente dal fatto che la tabella si trovi in una pagina di destra o di sinistra; i 14mm circa, che mancano alla tabella per arrivare alla larghezza della gabbia, potrebbero servire per la didascalia e per lo spazio che ci vuole fra questa e la tabella.

[1] Questi valori con sei cifre decimali, che usano il punto invece della virgola decimale, sono stati calcolati direttamente dal programma di compilazione; soffrono del fatto che il nucleo di TEX esegue i calcoli dietro le quinte con una matematica che gestisce solo numeri interi, quindi rappresentando i valori fratti delle lunghezze mediante gli *scaled points*, `sp`, cioè con numeri interi, dove $1\text{pt} = 2^{16}\text{sp}$. Versioni aggiornate del sistema TEX consentono di lavorare direttamente con le lunghezze in modo più efficiente, ma sempre soltanto con numeri interi. Usando le estensioni del linguaggio L3, sarebbe possibile eseguire calcoli in virgola mobile.

Si vede subito che i limiti alla tabella sono legati non solo alla gabbia del testo, ma anche al font usato e al suo corpo. È noto che il font Polatino, a pari corpo, è più largo del font Latin Modern, usato in questa guida, e decisamente più largo del font Times (in una qualunque delle sue realizzazioni); è evidente che la stessa tabella composta in Palatino, potrebbe uscire fuori dalla gabbia del testo, mentre potrebbe esservi contenuta comodamente se fosse composta in Times. Non scendiamo in ulteriori dettagli relativi ai font, perché la buona tipografia, senza un motivo specifico, non esegue cambiamenti di font nello stesso documento: talvolta cambia il corpo, ma non la collezione di font stilisticamente coerenti.

È ovvio che il massimo numero di colonne dipende anche dal contenuto più largo delle celle di ciascuna colonna. Bisogna anche tenere conto dello spazio intercolonna e dello spazio a sinistra della prima e di quello a destra dell'ultima colonna.

2.1 Tabelle troppo larghe

Non ci si può lamentare del fatto che una certa colonna non stia nella pagina o, più semplicemente nella gabbia, perché significa che non si è fatta un''adeguata programmazione della tabella; o meglio, se la tabella è solo un poco sporgente fuori della gabbia ci sono diversi modi per aggiustare le cose, ma non si può comporre una tabella in corpo "`\tiny`", solo perché la tabella è larga quasi il doppio della larghezza della gabbia.

In questi casi si può rimediare nei modi seguenti, che generalmente non sono mutuamente esclusivi.

1. La prima cosa che viene in mente è quella di ridurre il corpo del 10% o del 20% (il 20% può essere già troppo); basta racchiudere la tabella dentro un gruppo nel quale sia specificato un font di corpo `\small` oppure `\footnotesize`; siccome quasi sempre una tabella è resa flottante se si trova dentro un ambiente *table*, l'ambiente stesso forma il gruppo e la dichiarazione di corpo vi si esaurisce dentro. Per esempio:

```
\begin{table}\small
... % qui ci vuole \caption se la didascalia è sopra la tabella
\begin{tabular}{...}
...
```

```
\end{tabular}
... % qui ci vuole \caption se la didascalia è sotto la tabella
\end{table
```

2. La cosa è leggermente più complessa se si vuole comporre una tabella non flottate, perché bisogna creare il gruppo. Il modo più semplice consiste nel usare un ambiente *center* invece dell'ambiente `\table`, ricordando che senza l'ambiente flottante il comando `\caption` non funziona correttamente. Supponendo di voler mettere la didascalia sopra la tabella (come si fa solitamente nell'Europa continentale) la soluzione è la seguente:

   ```
   \begin{center}\small
     \captionof{table}[...]{...}
     \begin{tabular}{...}
     ...
     \end{tabular}
   \end{center}
   ```

   Il comando `\captionof` è disponibile tramite il pacchetto `capt-of`; si tratta di un pacchetto semplicissimo che contiene una sola riga di codice, ma senza questa piccolissima aggiunta, sarebbe impossibile .comporre didascalie per oggetti che non siano contenuti dentro oggetti flottanti. La sua sintassi è semplicissima replica quella di `\caption` ma contiene un argomento in più:

`\captionof{`⟨*tipo*⟩`}[`⟨*didascalia breve*⟩`]{`⟨*didascalia lunga*⟩`}`

   Il ⟨*tipo*⟩ serve per specificare la parola che compare nell'etichetta della didascalia: se si specifica `table` la didascalia viene composta nelle stesso modo con cui la compone `\caption` eseguito dentro un ambiente *table*.
3. Se la tabella sporge fuori dai margini di pochi millimetri, la si può inscatolare con `\makebox` a cui si specifichi una larghezza pari a `\linewidth` o a `\textwidth` secondo lo schema seguente:

   ```
   \makebox[\linewidth]{%
     \begin{tabular}{....}
     ...
     \end{tabular}
   }%
   ```

In questo modo, la tabella viene centrata e la sporgenza viene divisa equamente fra il margine sinistro e il margine destro.

4. Se la tabella sporge nel margine destro ma non esce dalla larghezza della pagina fisica, allora una soluzione abbastanza buona è quella di farla sporgere nel margine esterno; bisogna controllare il numero della pagina, perché il margine esterno è quello destro per le pagine dispari, ed è quello sinistro per le pagine pari. Quindi una soluzione, che ricorre ai comandi nativi di TEX, potrebbe essere la seguente:

```
\begin{table}
\caption[...]{...}
\hbox to\linewidth{\unless\ifodd\value{page}\hss\fi
\begin{tabular}{...}
...
\end{tabular}\ifodd\value{page}\hss\fi}
\end{table}
```

Si noti che il poco conosciuto comando `\unless` inverte l'esito del test, come è facile indovinare dal suo nome; `\hss` inserisce spazio elastico di larghezza naturale nulla, ma dotato di una buona dose di allungamento e accorciamento. La costruzione sintattica `\hbox to\linewidth` serve per creare una scatola orizzontale di larghezza specificata.
Si noti ancora che questo metodo non può essere usato quando si compone a due colonne, perché bisogna fare riferimento non al margine esterno della gabbia ma a quello della singola colonna; chi scrive non conosce nessun comando accessibile all'utente per controllare se l'oggetto flottante si trovi nella colonna di sinistra o in quella di destra.

5. Componendo a due colonne, si potrebbe ricorrere alle macro che servono per le note marginali, ma sebbene sia possibile, il codice risulta troppo complesso da descrivere in questa guida; chi scrive l'ha fatto, ma sebbene il tutto possa funzionare non è affatto garantito che faccia sempre proprio quello che si vorrebbe fare. Si tenga conto anche che gli oggetti flottanti non accettano note marginali e questo richiede di aggirare il problema con macro sibilline.
6. Secondo chi scrive, la soluzione migliore per rendere una tabella *leggermente* troppo larga compatibile con la giustezza della gabbia del testo consiste nel ricorrere al comando `\resizebox` definito dal

pacchetto `graphicx`; siccome si tratta di un pacchetto praticamente sempre già caricato per la composizione della stragrande maggioranza dei documenti, esso è virtualmente già disponibile. Volendo, si può anche combinarlo con il comando `\rotatebox` per ruotare di 90° e scalare una tabella troppo larga ma non più dell'altezza della gabbia del testo. Il codice da usare è del tipo seguente, da aggiustare a seconda delle specifiche esigenze richieste per una data tabella.

```
\begin{table}
  \caption[...]{...}
  \resizebox{\linewidth}{!}{\begin{tabular}{...}
  ...
  \end{tabular}%
}\end{table}
```

Il punto esclamativo che compare nel secondo argomento del comando `\resizebox` serve per scalare in altezza il suo contenuto esattamente nella stessa proporzione usata per scalarlo in larghezza; la tabella quindi non viene deformata nel suo aspetto grafico.

Sembra strano aver cominciato questa guida con i problemi da affrontare a tabella composta, invece di cominciare con la descrizione di come si creano le tabelle. Ma è intenzionale, proprio per sottolineare l'importanza di progettare la tabella prima di crearla, proprio per poter evitare in anticipo i problemi descritti.

2.2 Troppe colonne

Un errore che viene commesso spesso è quello di cercare costruire tabelle con troppe colonne, o con poche colonne troppo larghe.

Come si fa prima ancora di avere composto la tabella a capire che ci sono troppe colonne e che alcune (o tutte) sono troppo larghe?

Innanzi tutto bisogna avere un'idea precisa delle dimensioni della gabbia del testo; senza questa informazione, anche approssimata, non è possibile fare nessuna stima.

Per questo scopo si usino i comandi seguenti:

```
\showthe\textwidth
\showthe\textheight
```

all'interno del testo del documento. Compilando il documento, vengono mostrati sulla console di compilazione (o se sfuggisse vengono scritte anche nel file `.log`) le due dimensioni richieste espresse in punti tipografici. La stima si fa facilmente, perché 3pt ≈ 1mm.[2]

Per impostazione predefinita, il contenuto di ogni cella è circondato a destra e a sinistra da uno spazio di almeno 6pt ≈ 2mm; basta quindi stimare la lunghezza del contenuto più largo nelle celle di una stessa colonna, aggiungervi 4mm e così si ha una buona stima della larghezza dell'intera colonna. Più difficile è stimare la larghezza del contenuto più largo; non è difficile quando la colonna è di tipo `p` o `m` o `b`, perché la larghezza della colonna è specificata nell'argomento del suo descrittore; invece con le colonne di tipo `l`, `c` e `r`, l'operazione è un può più complessa; una buona stima si può ottenere facendosi stampare dal programma stesso il valore di 1ex all'interno di un brano di testo nel quale sia in vigore lo stesso font usato per comporre quel contenuto; il comando da usare è un po' più complesso:

```
{\dimen0=1ex\showthe\dimen0}
```

In questo testo si ha che 1ex vale `4.71944`pt. Con buona approssimazione la larghezza di un testo di n lettere è circa pari a $n$ex; per esempio, la parola 'lunghezza' è di 9 lettere, quindi la sua lunghezza nel font corrente è stimabile in $9 \times 4,72\text{pt} \approx 42,5\text{pt}$, mentre la sua lunghezza vera è di `46.84401`pt.[3] Sì, l'errore è circa del 10%, ma è sempre una stima accettabile.

Fatte queste stime per ciascuna colonna, aumentate di 4mm per il numero delle colonne, non è difficile disporre di una decente approssimazione della larghezza della tabella che si vorrebbe comporre.

Bisogna sempre fare questi calcoli e queste stime per ogni tabella che si vuole comporre? No, certamente non è necessario; se la tabella è ben progettata, si dispone già di una buona certezza che non ci saranno problemi. Eppure, talvolta sul forum del GUIT compaiono domande del tipo: "Ho questa tabella che devo inserire nella tesi, composta con questa classe; è troppo

[2] Per l'esattezza il coefficiente di proporzionalità fra i punti e i millimetri vale (72,27pt/pollice)/(25,4mm/pollice) ≈ 2,84528pt/mm. Arrotondare al valore 3 vuol dire accettare un'approssimazione del 5%.

[3] Tutti i valori numerici, riportati con sei cifre decimali, sono calcolati dal programma stesso; non necessariamente sono "esatti" (per il motivo spiegato in precedenza), ma sono quelli che il programma usa.

grande ed esce dal foglio; che cosa posso fare?”. Si va a vedere il codice della tabella e si scopre che ha 20 colonne ciascuna con una intestazione larghetta; ma anche solo gli spazi attorno a ogni contenuto ammontano a $20 \times 4\text{mm} = 80\text{mm}$, quindi già i soli spazi attorno alle colonne occupano due terzi della giustezza; figuriamoci quanto sarebbe larga l’intera tabella se si tenesse conto anche dell’effettivo contenuto dei termini più larghi di ciascuna colonna, presumibilmente le intestazioni.

Nel progettare la tabella si tenga presente che spesso, invece di comporla per colonne, si può optare per una composizione per righe. Le 20 colonne della tabella dell’esempio precedente diventano 20 righe e le precedenti righe diventano colonne; la riga delle intestazioni diventa una colonna, e di colonne larghette ne rimane una sola.

Gli americani hanno un un motto molto diffuso: *Think ahead*, “Pensaci prima”. Ecco è questo ciò che bisogna fare: pensarci prima, pensare prima di agire.

Gli ambienti *tabular* e *tabular** 3

Progettata una tabella, bisogna comporla. Cominciamo con gli ambienti *tabular* e *tabular**; si premette subito che quanto si dice a proposito di *tabular* vale anche per *tabular**.

Va subito detto che tutti gli ambienti, non solo quelli descritti qui, si fondano sull'uso intelligente del comando di basso livello `\halign`; esso è contenuto nel linguaggio nativo di TEX, serve per allineare delle celle orizzontalmente e per impilarle verticalmente; LATEX si limita ad aggiungere una buona interfaccia utente, per superare le difficoltà intrinseche con la sintassi dei comandi nativi. Il kernel di LATEX, quindi, definisce solo gli ambienti *array*, *tabular* e *tabular**; in realtà, nel kernel viene definito anche l'ambiente *tabbing*, che però non forma una tabella ma una tabulazione. In linea di massima una tabulazione si può fare anche con *tabular* o *tabular**, forse anche in modo più agevole; il vantaggio di *tabbbing* è che si può inserire in qualunque punto del testo, senza bisogno di rendere flottante la tabulazione, perché consente un fine pagina in qualunque punto tra le righe della tabulazione. Forse il paragone migliore si potrebbe fare con *longtable*. Ma qui si esula un po' troppo dal tema di questo capitolo.

Non esula invece l'informazione che l'ambiente *array* è descritto meglio nella guida tematica *Regole e consigli per comporre la matematica delle scienze sperimentali*, dedicata essenzialmente alla matematica, anche se vi aggiunge le speciali regole per comporre la matematica delle grandezze. Quindi, qui se ne accennerà un poco, ma si rinvia il lettore ad approfondire la questione su quella guida; essa è reperibile nella sezione Documentazione del forum del GUIT.

Tranne *array*, che va usato solo in modo matematico, tutti gli altri ambienti vanno o andrebbero usati sempre all'interno dell'ambiente di flottaggio *table*, a meno che non se ne dica esplicitamente il contrario.

O meglio: piccole tabelle che non si sviluppano molto in senso verticale possono anche essere messe nel testo, magari dentro un ambiente *center*; oppure piccole tabelle che non si sviluppano molto né in senso verticale né in quello orizzontale possono essere collocate parzialmente nel margine, con o senza didascalia, ricorrendo all'ambiente *wraptable*.[1] Chi scrive ha usato molto raramente quest'ultima soluzione, e solo in casi molto particolari; segnala questa possibilità, ma la rinvia alla progettazione della tabella, perché la collocazione entro un capoverso, e parzialmente sporgente nel margine, presenta dei problemi; si tenga presente che l'ambiente *wraptable* consente l'uso di parametri di collocamento con i quali si può decidere se rendere flottante il suo contenuto, oppure se mantenerne fissa la posizione.

3.1 L'ambiente *tabular*

L'ambiente *tabular* richiede un comando di apertura particolare; infatti la sua sintassi è la seguente:

```
\begin{tabular}[⟨allineamento verticale⟩]{⟨descrittori delle colonne⟩}
⟨prima cella⟩ & ⟨seconda cella⟩ & ... &... \\
⟨prima cella⟩ & ⟨seconda cella⟩ & ... &... \\
...
⟨prima cella⟩ & ⟨seconda cella⟩ & ... &... \\
\end{tabular}
```

3.1.1 I codici di allineamento verticale

È raro che si debba usare l'argomento facoltativo ⟨*allineamento verticale*⟩ ma è utile disporne quando si deve allineare la tabella con il testo circostante; oppure quando si devono allineare tabelle affiancate; oppure quando si deve annidare un *tabular* dentro una cella di un altro *tabular*. In ogni caso i codici per la posizione verticale sono i seguenti.

`t` La prima riga della tabella viene allineata con la prima riga del testo circostante o con quella di una tabella adiacente.

`b` L'ultima riga della tabella viene allineata con l'unica o l'ultima riga del testo circostante o di una tabella adiacente.

[1] Il pacchetto `wrapfig` definisce gli ambienti *wrapfigure* e *wraptable*.

`c` L'asse matematico della tabella viene allineato con l'asse matematico del testo circostante o di una tabella adiacente. Si ricorda che l'asse matematico è una linea orizzontale ideale che passa a metà fra le due linee che formano l'operatore di relazione `=`. Esso non è necessariamente all'altezza di $0,5$ex, cioè passante per l'incrocio delle due linee oblique che formano il disegno della lettera 'x', ma è una caratteristica di disegno di ogni font. La posizione `c` è quella predefinita.

Il codice seguente mostra l'effetto di questi codici di posizione con una piccola matrice in linea col testo:

```
La posizione della tabella \begin{tabular}{@{}l}Giulio @{}\\
Cesare\end{tabular} è allineata di default con l'asse
matematico. Con il codice di posizione  't'
\begin{tabular}[t]{@{}l@{}}Giulio \\ Cesare\end{tabular} ha
la prima riga allineata col testo circostante. Con il
codice 'b' la seconda riga \begin{tabular}[b]{@{}l@{}}Giulio \\
Cesare\end{tabular} è allineata con il testo circostante.
```

Infatti si ottiene:

La posizione della tabella $\begin{array}{l}\text{Giulio}\\ \text{Cesare}\end{array}$ è allineata di default con l'asse matematico. Con il codice di posizione 't' $\begin{array}{l}\text{Giulio}\\ \text{Cesare}\end{array}$ ha la prima riga allineata col testo circostante. Con il codice 'b' la seconda $\begin{array}{l}\text{Giulio}\\ \text{Cesare}\end{array}$ è allineata con il testo circostante.

3.1.2 I DESCRITTORI DI COLONNA

Il kernel di LaTeX prevede solo certi codici che permettono di definire le varie colonne e i loro separatori in modo simbolico. Questi codici sono un po' criptici e alcuni sono poco mnemonici, tuttavia con un po' di pratica non ci sono problemi ad usare i più comuni. Nessuna lista di descrittori è predefinita, ma deve essere sempre specificata dall'utente; essa è formata dalla loro sequenza con o senza spazi per separarli; alcuni vogliono un argomento obbligatorio, altri non ne hanno bisogno.

`c` Specifica che il contenuto della cella deve essere centrato (c: *center*) nello spazio disponibile. Un eventuale testo di una certa lunghezza non viene ripiegato su più righe.

`l` Specifica che il contenuto della cella deve essere allineato a sinistra (l: *left*) nello spazio disponibile. Anche in questo caso, un eventuale testo di una certa lunghezza non viene ripiegato su più righe.

`r` Specifica che il contenuto della cella deve essere allineato a destra (r: *right*) nello spazio disponibile. Anche in questo caso, un eventuale testo di una certa lunghezza non viene ripiegato su più righe.

`p{`⟨*larghezza*⟩`}` Specifica una cella in cui trova posto un capoverso (p: *paragraph*) avente una giustezza pari a ⟨*larghezza*⟩; la prima riga del capoverso viene allinea all'unica o alla prima riga delle celle adiacenti.

`@{`⟨*intercolonna*⟩`}` Definisce una *@-espressione*, cioè un codice qui chiamato ⟨*intercolonna*⟩ formato di una sequenza di comandi specifici per definire che cosa va messo fra una colonna e l'altra; l'*@-espressione* sostituisce completamente quanto si trova fra i contenuti di celle adiacenti; l'⟨*intercolonna*⟩ può anche essere formato da niente, cioè le due graffe devono esserci lo stesso, perché @ richiede un argomento obbligatorio, ma questo può essere tranquillamente vuoto; lo si è fatto esplicitamente nella descrizione dei codici di posizione esemplificati nella pagina 23.

`|` Serve per introdurre un filetto verticale al centro dello spazio che separa due colonne adiacenti, oppure a sinistra della prima cella di ogni riga, oppure a destra dell'ultima cella di ogni riga. Però, vedi più avanti il paragrafo 3.1.5.

`\vline` Sostituisce il descrittore `|` quando il filetto deve essere indicato in una *@-espressione*. Di nuovo vedi più avanti il paragrafo 3.1.5.

`\extracolsep{`⟨*larghezza*⟩`}` Serve dentro le *@-espressioni* per specificare che, dal separatore successivo a quello dove si trova il comando, bisogna aggiungere lo spazio indicato mediante ⟨*larghezza*⟩; questa può essere una lunghezza esplicita assoluta, per esempio 2mm; oppure una lunghezza specificata mediante unità di misura dipendenti dal corpo del font in uso, per esempio `0.5`em; oppure uno spazio elastico, come ad esempio `\fill`.

`\fill` Rappresenta uno spazio elastico da inserire nell'argomento del comando `\extracolsep`. Non bisogna confondere questo spazio con quello che si ottiene in orizzontale con `\hfill`, o in verticale con

`\vfill`; questi sono comandi che spaziano; quello è solo la misura di uno spazio elastico, ma non è lui quello che spazia.

`*{`⟨*numero*⟩`}{`⟨*descrittori*⟩`}` Serve come abbreviazione per non dover indicare a mano una specifica serie di ⟨*descrittori*⟩ ripetuta ⟨*numero*⟩ volte; i ⟨*descrittori*⟩ possono a loro volta contenere altre espressioni abbreviate con `*`, ma se si supera il secondo livello di annidamento l'espressione complessiva diventa illeggibile. Questo descrittore è molto comodo quando si devono ripetere le stesse sequenze di descrittori più volte; per esempio, `*{12}{c}` indica una serie di dodici colonne tutte col descrittore `c`. Altro esempio: `*{5}{r@{,}l}` indica di ripetere 5 volte coppie di due colonne separate solo da una virgola, e senza altri spazi di separazione; la colonna di sinistra ha il suo contenuto allineato a destra, al contrario, la colonna di destra ha il suo contenuto allineato a sinistra; potrebbe servire per comporre una "colonna" di numeri fratti incolonnati rispetto alla virgola, nella colonna di sinistra si scrive la parte intera e in quella di destra la parte fratta. Questo è solo un semplice esempio, *da non usare*, perché con il pacchetto `siunitx` viene definito un nuovo tipo di colonna, `S` che ha quello scopo ed è più semplice e più versatile da usare.

I prossimi tre comandi non sono descrittori di colonne, ma interferiscono a ragion veduta con la composizione della tabella. Il primo interrompe la composizione di una riga della tabella, anche se non contiene le celle di tutte le colonne; il secondo inserisce sotto tutte le celle di una riga un filetto orizzontale; il terzo inserisce un filetto orizzontale solo sotto le celle di alcune colonne.

`\\[`⟨*altezza*⟩`]` È il solito comando di fine riga, simile a quello che si può usare nel testo; bisogna usarlo con cautela quando l'ultima cella di una riga è una cella di tipo `p` (o `m` o `b`; per questi due descrittori si veda il prossimo paragrafo) perché il suo uso diventa ambiguo: è un comando di fine riga per il capoverso contenuto nella cella, o è il fine riga della fila di celle della tabella? In questo caso è preferibile usare sempre i comando `\newline` e, comunque, è sempre opportuno racchiudere il capoverso della cella in un gruppo. Si veda, però, il prossimo paragrafo per vedere altre soluzioni.

`\hline` Traccia un filetto orizzontale attraverso tutta la tabella. Sarebbe opportuno non fare uso di questo tipo di filetti, salvo pochissime situazioni gestite meglio dal pacchetto `booktabs` (vedi più avanti la sezione 3.1.5).

`\cline{`⟨col_1⟩`-`⟨col_2⟩`}` traccia un filetto orizzontale attraverso parte della tabella, dalla colonna ⟨col_1⟩ alla colonna ⟨col_2⟩ comprese. È bene non servirsi di questi filetti, che sono gestiti meglio con il pacchetto `booktabs`.

3.1.3 Ulteriori descrittori e modificatori

Il pacchetto `array` introduce molte utili estensioni all'ambiente *tabular* restando compatibile anche con diversi altri ambienti. Meritano di essere segnalate queste estensioni.

`m{`⟨*larghezza*⟩`}` Descrive una colonna di celle come nel tipo `p`, con la differenza che l'allineamento del contenuto con le celle adiacenti viene fatto sulla base dell'asse matematico (m: *middle*).

`b{`⟨*larghezza*⟩`}` Simile al precedente descrittore, ne differisce solo nel fatto che allinea la linea di base (b: *bottom*) con quella delle celle adiacenti.

`>{`⟨*dichiarazioni*⟩`}` premesso a qualunque altro descrittore, fa in modo che la lista delle ⟨*dichiarazioni*⟩ venga inserita all'inizio di ogni cella della colonna; per esempio, se una tale lista contenesse solo la dichiarazione `\itshape`, tutte le celle in quella colonna verrebbero composte in corsivo. Fra le dichiarazioni ci possono stare anche i comandi di apertura di un'ambiente matematico, per esempio `$`, ma poi bisogna ricordarsi che l'ambiente va chiuso con il prossimo descrittore. Come caso particolare, l'ultimo descrittore della lista potrebbe essere un comando che accetta argomenti; ma questo implica che il contenuto di ogni cella deve contenere lo stesso numero di token, quanti ne sono richiesti dal comando, o come token isolati (per esempio, singole lettere, oppure come argomenti racchiusi fra graffe). La sintassi, benché offra questa possibilità, è molto critica e rende delicata la scrittura dei contenuti delle celle, quindi è meglio evitare i comandi che richiedano più di un argomento, e possibilmente che questo sia un unico token.

`<{`⟨*dichiarazioni*⟩`}` posposto a qualunque descrittore di colonna, appende la sua lista di ⟨*descrizioni*⟩ alla fine del contenuto della cella; ovviamente, nessuna delle dichiarazioni di questa lista può essere un comando che richiede argomenti.

`!{`⟨*intercolonna*⟩`}` Si comporta come il descrittore `@` ma, a differenza di questo, non sostituisce gli spazi adiacenti; quindi, in un certo senso agisce come `|`, solo che al posto di un filetto, inserisce quanto specificato in ⟨*intercolonna*⟩.

`w{`⟨*allinea*⟩`}{`⟨*larghezza*⟩`}` crea le celle come se contenessero tutte il comando `\makebox[`⟨*allinea*⟩`][`⟨*larghezza*⟩`]{`⟨*contenuto della cella*⟩`}`. Se il ⟨*contenuto della cella*⟩ è più largo di ⟨*larghezza*⟩, esso fuoriesce dalla cella e può andare a ricoprire le celle adiacenti. Si noti che se si specifica `w{l}{`⟨*larghezza*⟩`}`, si ottiene una colonna simile a una colonna descritta da `l`, ma con le celle tutte della ⟨*larghezza*⟩ specificata; similmente il descrittore `w` con ⟨*allinea*⟩ che vale `c` o `r` si comporta in modo simile, quindi come colonne di tipo `c` o `r` ma di larghezza prefissata. Questo aiuta molto l'utente, per esempio, a comporre tabelle con colonne tutte della stessa larghezza.

`W{`⟨*allinea*⟩`}{`⟨*larghezza*⟩`}` si comporta come il descrittore `w`, salvo che, se il ⟨*contenuto della cella*⟩ è più largo di ⟨*larghezza*⟩, emette un messaggio d'errore.

`\newcolumntype{`⟨*lettera*⟩`}>{`⟨*dichiarazioni*⟩`}d<{`⟨*altre dichiarazioni*⟩`}`
Definisce un nuovo tipo di colonna mediante una sola lettera maiuscola o minuscola purché sia diversa da qualunque altro descrittore `d` in vigore nell'ambiente in cui si sta lavorando; vale a dire diverso dai descrittori presenti di default, o creati dai pacchetti caricati, o da comandi `\newcolumntype` specificati nel preambolo del documento, o da quelli definiti localmente all'interno, per esempio, di un ambiente *table*. Un esempio:

```
\newcolumntypeC{>{\centering$}c<{$}}
```

definisce un nuovo tipo di colonna in cui tutte le celle hanno contenuto matematico centrato. I nuovi tipi di colonna possono essere usati a loro volta dentro le definizioni di altri nuovi tipi di colonna.[2]

[2] Ovviamente, un nuovo tipo di colonna non può essere definito con una espressione che contenga lo stesso tipo; in altre parole quando un tipo di colonna è stato definito,

`\extrarowheight{`⟨*altezza*⟩`}` Aggiunge ⟨*altezza*⟩ al pilastrino della prima cella di ogni riga di una tabella senza toccarne la profondità, in modo che se esistono filetti orizzontali `\hline` questi non tocchino le maiuscole eventualmente presenti anche in una sola cella di ogni riga. Si noti la differenza rispetto a `\arraystretch`, che cambia sia l'altezza sia la profondità del pilastrino; lo scopo è diverso ma esteticamente potrebbero essere incompatibili, perché i loro effetti si sommano e il risultato potrebbe non essere gradevole.

`\\` Come comando di fine riga della tabella si comporta nel solito modo, ma se l'ultima cella di una tabella contiene un capoverso, come si è già detto, il suo significato diventa ambiguo. Come fine riga di un capoverso si può usare `\newline` e come fine riga della tabella si può usare `\tabularnewline`. Meglio ancora, si può usare il prossimo comando.

`\arraybackslash` Questo è un comando che conviene usare fra quelli che si inseriscono nella lista di ⟨*dichiarazioni*⟩ di `>` da inserire nella personalizzazione dell'ultima colonna della tabella; di fatto rende il comando `\\` equivalente a `\tabularnewline`, in modo che usandolo come terminatore dell'ultima cella che contiene un capoverso, eventualmente caratterizzato da altre personalizzazioni come, per esempio, `\raggedright`, sia senz'altro interpretato come comando per la fine della riga della tabella.

Va da sé che alcuni di questi comandi, ed altri non citati qui, vanno usati correttamente e, quindi, la lettura della documentazione del pacchetto `array` è molto importante.

Nello stesso tempo, è bene ricordare che il kernel definisce un altro comando, che permette di definire singole celle che ne occupano un certo numero nelle tabelle descritte. La sintassi è la seguente.

`\multicolumn{`⟨*numero colonne*⟩`}{`⟨*descrittore*⟩`}{`⟨*contenuto della cella*⟩`}`

dove il ⟨*numero colonne*⟩ è un numero maggiore o uguale a 1 e minore o uguale al numero totale delle celle di una riga nella tabella. Va da sé che se si usano diversi comandi `\multicolumn` nella stessa riga, la somma dei loro numeri e del numero delle celle non sostituite da questi comandi deve

non lo si può più modificare.

essere uguale al numero di celle di una riga della tabella complessiva. Il ⟨*descrittore*⟩ è un descrittore letterale fra quelli descritti in questo e nel paragrafo precedente, ma deve essere seguito dallo stesso tipo di spaziatura che segue l'ultima cella sostituita. Solo sostituendo la prima cella di una riga, il descrittore letterale può essere preceduto dal primo spaziatore della prima cella di ogni riga della tabella complessiva. Queste celle multicolonna servono essenzialmente per le intestazioni delle colonne prese singolarmente o a gruppi.

Per esempio, se la tabella comincia con un'apertura del tipo:

```
\begin{tabular}{|*5l|rr|}
```

vuol dire che si è definita una tabella di 7 colonne, di cui le prime cinque hanno gli allineamenti a sinistra, e sono complessivamente racchiuse fra due filetti verticali, e sono seguire da altre due colonne allineate a destra, delimitate a destra da un filetto verticale. L'intestazione delle prime cinque può essere, per esempio, "Anni scolastici" scritto in neretto, e le ultime due potrebbero essere intestate "Esami". La prima intestazione, ricordando che i parametri costituiti da un solo token non hanno bisogno di essere racchiusi fra graffe, potrebbe essere:

```
\multicolum5{|>{\bfseries}c|}{Anni scolastici}
```

e l'intestazione delle ultime due colonne potrebbe essere

```
\multicolumn2{>{\bfseries}c|}{Esami}
```

In definitiva, le prime righe della tabella potrebbero essere:

```
\begin{tabular}{|*5l|rr|}
\hline
\multicolum5{|>{\bfseries}c|}{Anni scolastici}&
           \multicolum2{>{\bfseries}c|}{Esami}\\
\hline
```

Ripeto: il lettore non si stupisca se gli argomenti del comando `\multicolumn` non contengono tutte le graffe che appaiono nella sua sintassi; infatti se l'argomento è costituito da un solo segno, un solo token, le graffe sono superflue, si potrebbero eliminare anche le graffe a cavallo di `\bfseries`, pur di far seguire questo token complesso da uno spazio.

Il lettore non si stupisca nemmeno se la tabella contiene dei filetti verticali e orizzontali creati con lo spaziatore `|` e con i comandi `\hline`, quando si è detto che i filetti verticali e orizzontali sono da evitare e che bisogna invece affidarsi alle funzionalità fornite dal pacchetto `booktabs`; si vedrà più avanti come fare, ma qui si voleva porre l'attenzione sui descrittori e spaziatori generali e quelli delle multicolonne.

3.1.4 Personalizzazione delle tabelle

Lo stile con cui vengono composte le tabelle all'interno di *tabular*, e degli altri ambienti compatibili, sono date da un certo numero di parametri che vanno usati per la personalizzazione; spesso si usa `\renewcommand` perché si tratta di macro invece che di lunghezze, o spazi elastici, o altro. Eccone l'elenco.

`\arraycolsep` È una lunghezza e può essere reimpostata con `\setlength` o con una semplice assegnazione, per esempio, `\arraycolsep=3mm`; il suo valore rappresenta metà dello spazio fra le colonne di una matrice e serve, quindi per l'ambiente *array*. La versione di *tabular* da usare in modo matematico; il suo valore preimpostato è 5pt.

`\tabcolsep` È una lunghezza e può essere reimpostata sia con `\setlength` sia con una semplice assegnazione, per esempio, `\tabcolsep=1mm`; il suo valore preimpostato vale 6pt. Questo è metà dello spazio intercolonna fra le colonne di una tabella composta con *tabular* o uno dei suoi simili.

`\arrayrulewidth` È una lunghezza e può essere reimpostata come indicato per le precedenti lunghezze; essa rappresenta lo spessore dei filetti `\hline` `\vline` e `|`; il suo valore preimpostato vale 0,4pt.

`\doublerulesep` È lo spazio verticale che separa due filetti `\hline` consecutivi; il valore preimpostato è 2pt. Lo si può reimpostare nella stessa maniera indicata per le lunghezze precedenti.

`\arraystretch` Nonostante il nome faccia pensare all'ambiente *array*, questa macro funziona anche con l'ambiente *tabular* e i suoi simili. È una macro il cui argomento serve per variare per un fattore di scala, indicato dal suo argomento (preimpostato al valore `1.0`), le dimensioni (altezza e profondità) del pilastrino che gli ambienti inseriscono nella prima cella di ogni riga; questo pilastrino indica l'altezza *minima* di

ogni riga della tabella o della matrice; se il contenuto di una cella è più alto e/o più profondo, la riga assume le dimensioni di questa cella. Se si cambia aumentandolo il valore di `\arraystretch`, le righe della tabella o della matrice appaiono più distanziate; se lo si diminuisce, le righe possono assumere una distanza irregolare, perché questa viene determinata dal contenuto delle celle e non dal pilastrino; tuttavia è anche lecito il valore nullo, ma con il valore nullo bisogna essere sicuri di quello che si sta facendo, pena una tabella o una matrice orrenda.

Merita ricordare, comunque, che tutto quanto detto in merito ai filetti `\hline`, `\vline` e `|` non è da prendere in troppa considerazione, perché questi filetti, di cui si abusa per imitazione con ciò che si ottiene con i soliti word processor, non dovrebbero mai essere usati nelle tabelle ben formate, se non in rarissimi casi e in poche posizioni.

3.1.5 Personalizzazione dei filetti

Infatti, esiste il pacchetto `booktabs` che spiega perché i filetti verticali non si dovrebbero mai usare mentre quelli orizzontali si possono usare, ma con estrema parsimonia, al punto che vi si afferma che il numero massimo di filetti orizzontali che attraversano tutta la tabella è tre!

A maggior ragione, il pacchetto definisce solo tre nuovi filetti.

`\toprule` Serve per mettere un filetto leggermente più scuro del normale prima della prima riga della tabella, che spesso contiene le celle di intestazione delle colonne; provvede a un spazio verticale, in modo che il filetto non tocchi le lettere maiuscole. Può essere usato solo prima del contenuto della prima cella della tabella.

`\midrule` Serve per mettere un filetto sottile sotto la riga delle intestazioni; contiene sia uno spazio sopra sia uno spazio sotto per essere sufficientemente distanziato dai discendenti delle celle della riga di intestazione, sia dagli ascendenti delle celle della riga successiva della tabella; può quindi essere usato solo subito dopo il comando di fine riga della riga di intestazione.

`\bottomrule` Serve per mettere un filetto leggermente più scuro sotto l'ultima riga della tabella; evidentemente può essere usato solo dopo il comando di fine riga dell'ultima riga della tabella

`\cmidrule[`⟨*spessore*⟩`](`⟨*trim*⟩`){`⟨*colonne*⟩`}` svolge più o meno lo stesso scopo del comando `\cline` dell'ambiente originale *tabular*, ma è più versatile. Il significato degli argomenti è il seguente.

1. L'argomento facoltativo ⟨*spessore*⟩ definisce lo spessore del filetto; tutte la grandezze dimensionali del pacchetto `booktabs` sono proporzionate alle caratteristiche dimensionali dei font in uso; per impostazione predefinita lo ⟨*spessore*⟩ vale 0,03em, quindi componendo con un corpo normale di 10pt lo spessore è *circa*[3] uguale a 0,3pt.
2. L'argomento facoltativo ⟨*trim*⟩ racchiuso fra parentesi *tonde* permette di tagliare via un pezzetto di filetto a destra e/o a sinistra; la sintassi è un po' particolare perché va specificato con i codici letterali `l` e/o `r` accostati e senza spazi o altri separatori in mezzo; ognuno di questi codici accetta un argomento facoltativo racchiuso fra parentesi graffe e, se questo argomento non è specificato, il suo valore di default vale 0,5em; quindi la scrittura completa è `l{`⟨$trim_1$⟩`}r{`⟨$trim_2$⟩`}`; sono utili da usare quando ci sono due `\cmidrule` consecutivi (accostati in orizzontale), oppure quando si trovano nella prima o nell'ultima cella, ai bordi della tabella, nel caso che anche la sporgenza dei filetti superiore, intermedio e inferiore siano loro stessi privi della sporgenza sopra la prima o l'ultima cella di una riga della tabella. Per i dettagli si rimanda alla documentazione del pacchetto `booktabs` che mostra anche un piccolo numero di esempi.
3. L'argomento obbligatorio ⟨*colonne*⟩ è costituito da due indici di colonna, anche coincidenti (per sopra- o sottolineare una sola cella) di cui il secondo non è inferiore al primo; sono separati da un trattino; se ne vedono due esempi nella tabella 3.1 il cui codice comincia nella pagina 33.

Si confrontino le due tabelle affiancate nella tabella 3.1. La tabella (*a*) è stata ricopiata dal manuale di Leslie Lamport, *LaTeX: A document preparation system – User guide and reference manual*, mentre la seconda (*b*) è

[3] La parola 'circa' è evidenziata perché, nonostante l'unità `em` abbia un nome che la metta in relazione con la lettera 'M' del font corrente, l'analisi delle proprietà metriche dei font è raramente uguale alla larghezza di quella lettera; in questa nota abbiamo 1em = `9.252pt`, mentre la larghezza della 'M' vale `4.72499`pt.

TABELLA 3.1 Due tabelle a confronto: (*a*) tabella costruita secondo il modello del kernel di LaTeX; (*b*) tabella costruita secondo il modello di `booktabs`.

Gnats	gram	$13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	9.99

(*a*)

Item		
Animal	Description	Price ($)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	9.99

(*b*)

tratta dalla documentazione di `booktabs`. La differenza è enorme, non solo per la presenza di due righe di intestazione, ma proprio per la mancanza dei filetti; la seconda è decisamente più leggibile, grazie all'assenza dei filetti.

Anche il codice è più semplice; qui lo si riporta a titolo di esempio.

```
\begin{table}
\caption[Due tabelle a confronto]{Due tabelle a confronto:
$(a)$ tabella costruita secondo il modello del kernel di \LaTeX;
$(b)$ tabella costruita secondo il modello di \pack{booktabs}.}
\label{tab:due-tabelle}
\begin{minipage}[t]{0.467\linewidth}\raggedright
  \begin{tabular}[b]{||l|lr||}
    \hline
    Gnats     & gram    & \$13.65   \\
    \cline{2-3}
              & each    &    0.01   \\
    \hline
    Gnu       & stuffed &   92.50   \\
    \cline{1-1}         \cline{3-3}
    Emu       & stuffed &   33.33   \\
    \hline
    Armadillo & frozen  &    9.99   \\
    \hline
  \end{tabular}
```

```
  \makebox[\hsize]{$(a)$}
\end{minipage}
\hfill
\begin{minipage}[t]{0.523\linewidth}\raggedleft
  \begin{tabular}[b]{llr}
    \toprule
    \multicolumn2c{Item}    &             \\
    \cmidrule{1-2}
    Animal    & Description & Price (\$)  \\
    \midrule
    Gnat      & per gram    & 13.65       \\
              & each        &  0.01       \\
    Gnu       & stuffed     & 92.50       \\
    Emu       & stuffed     & 33.33       \\
    Armadillo & frozen      &  9.99       \\
    \bottomrule
  \end{tabular}

  \makebox[\hsize]{$(b)$}
\end{minipage}
\end{table}
```

Si sottolinea l'importanza di non usare filetti verticali; di usare il minimo di filetti orizzontali; è così che vengono composte le tabelle in modo professionale. Il fatto che siano disponibili i mezzi per inserire filetti orizzontali e verticali a piacere non autorizza a farlo, anche se esistono casi particolari che richiedono alcuni di questi filetti. Potendo è meglio astenersi; se per la comprensione di una tabella complicata è necessario qualche filetto in più del minimo necessario, li si metta, dopo una ragionevole analisi dei pro e dei contro ,avendo sempre in mente il lettore.

3.1.6 Alcuni esempi

Si sono già mostrati alcuni esempi; qui si dà per scontato che oltre ai comandi del kernel di LaTeX si siano caricati anche i pacchetti `array` e `booktabs` in modo da poter mostrare alcune loro caratteristiche che sono talmente utili, che chi scrive li carica e li usa sempre.

TABELLA 3.2 La famiglia Rossi; tabella naturale

Nome	Ruolo	Età	Attività
Giovanni	padre	47	impiegato
Maria	madre	44	insegnante elementare
Giovanna	figlia	14	studente di scuola media
Piero	figlio	8	allievo di scuola elementare

Esempio 3.1 ***La composizione di una famiglia***

Mostriamo una tabella che riporta la composizione di una famiglia usando solo i comandi del kernel di LaTeX, cioè l'ambiente *tabular* e i comandi nativi che esso conosce. Per questo motivo si esamini la tabella 3.2 costruita con il codice seguente.

```
\begin{table}\centering % mai usare l'ambiente center
%                         dentro gli ambienti flottanti
\caption{La famiglia Rossi; tabella naturale}
  \label{tab:famiglia-1}
\begin{tabular}{llcl}\hline
Nome     &  Ruolo  & Età & Attività    \\
\hline
Giovanni &  padre  & 47  & impiegato   \\
Maria    &  madre  & 44  & insegnante elementare \\
Giovanna &  figlia & 14  & studente di scuola media \\
Piero    &  figlio &  8  & allievo di scuola elementare \\
\hline
\end{tabular}
\end{table}
```

Come si vede, la tabella viene abbastanza bene, ma i filetti sono quasi rasenti agli ascendenti delle lettere minuscole e alla parte superiore delle maiuscole; aggiungendo la ridefinizione di `\arraystretch` all'inizio dell'ambiente *table*, la si può aggiustare senza dover ricorrere a un fattore di allargamento troppo alto. Ecco il codice a il relativo risultato nella tabella 3.3.

```
\begin{table}\centering % mai usare l'ambiente center
%                         dentro gli ambienti flottanti
\renewcommand\arraystretch{1.1}
```

Tabella 3.3 La famiglia Rossi; tabella allargata

Nome	Ruolo	Età	Attività
Giovanni	padre	47	impiegato
Maria	madre	44	insegnante elementare
Giovanna	figlia	14	studente di scuola media
Piero	figlio	8	allievo di scuola elementare

```
\caption{La famiglia Rossi; tabella allargata}
  \label{tab:famiglia-2}
\begin{tabular}{llcl}\hline
  Nome         &  Ruolo    & Età   & Attività    \\
\hline
  Giovanni &  padre  & 47  & impiegato   \\
  Maria    &  madre  & 44  & insegnante elementare \\
  Giovanna &  figlia & 14  & studente di scuola media \\
  Piero    &  figlio &  8  & allievo di scuola elementare \\
\hline
\end{tabular}
\end{table}
```

L'effetto di un allargamento solamente del 10% è piccolo ma l'occhio lo percepisce e la tabella è accettabile. Ma si può fare di meglio.

Proviamo allora a gestire lo spessore dei filetti, limitandone l'effetto al singolo filetto, ingrossiamo il primo e l'ultimo. Ecco il codice e il risultato finale nella tabella 3.4.

```
\begin{table}\centering % mai usare l'ambiente center
%                         dentro gli ambienti flottanti
\renewcommand\arraystretch{1.1}
\newcommand\thickhline{\hrule height \arrayrulewidth}
\caption{La famiglia Rossi; tabella con filetti spessi}
\begin{tabular}{llcl}
\noalign{\arrayrulewidth=1pt\thickhline}
  Nome     &  Ruolo  & Età & Attività    \\
\hline
  Giovanni &  padre  & 47  & impiegato   \\
```

TABELLA 3.4 La famiglia Rossi; tabella con filetti spessi

Nome	Ruolo	Età	Attività
Giovanni	padre	47	impiegato
Maria	madre	44	insegnante elementare
Giovanna	figlia	14	studente di scuola media
Piero	figlio	8	allievo di scuola elementare

```
  Maria    &  madre  & 44  & insegnante elementare \\
  Giovanna &  figlia & 14  & studente di scuola media \\
  Piero    &  figlio &  8  & allievo di scuola elementare \\
\noalign{\arrayrulewidth=1pt\thickhline}
\end{tabular}
\end{table}
```

Sì, ci siamo arrivati ma abbiamo dovuto usare comandi nativi di TEX e raramente documentati nelle normali guide di LATEX; da dove salta fuori `\noalign`? Probabilmente vuol dire che il suo argomento non deve essere trattato come materiale da allineare alle colonne, ma chi lo conferma? E quel `\hrule`, seguito da quella parola `height` che non è un comando, che cosa vuol dire? Dov'è documentato? Ah, benedetto il pacchetto `booktabs` che fa questo genere di cose dietro le quinte e ci esonera dal cercare questi comandi nativi, poco o per nulla conosciuti dall'utente normale.

Esempio 3.2 ***La stessa famiglia e*** `booktabs`

Usando le funzionalità di `booktabs` le cose sono decisamente più semplici; ecco il codice e il risultato nella tabella 3.5.

```
\begin{table}\centering % mai usare l'ambiente center
%                         dentro gli ambienti flottanti
\caption{La famiglia Rossi;
         tabella con filetti di \packstyle{booktabs}}
         \label{tab:famiglia-4}
\begin{tabular}{llcl}
\toprule
  Nome     &  Ruolo  & Età & Attività    \\
\midrule
```


Tabelle, versione v.1.1.02 del 2021-09-22

Tabella 3.5 La famiglia Rossi; tabella con filetti di `booktabs`

Nome	Ruolo	Età	Attività
Giovanni	padre	47	impiegato
Maria	madre	44	insegnante elementare
Giovanna	figlia	14	studente di scuola media
Piero	figlio	8	allievo di scuola elementare

```
  Giovanni &  padre  & 47  & impiegato   \\
  Maria    &  madre  & 44  & insegnante elementare \\
  Giovanna &  figlia & 14  & studente di scuola media \\
  Piero    &  figlio &  8  & allievo di scuola elementare \\
\bottomrule
\end{tabular}
\end{table}
```

Esempio 3.3 ***La famiglia Rossi con una colonna di capoversi***

Siccome le colonne dei capoversi sono relativamente strette, siamo "costretti" a usare le funzionalità del pacchetto `array` per dichiarare la quarta colonna come composta di capoversi, relativamente corti, e quindi con la composizione in bandiera allineata a sinistra, tabella 3.6.

```
\begin{table}\centering % mai usare l'ambiente center
%                         dentro gli ambienti flottanti
\caption{La famiglia Rossi;
         tabella con colonna \descrittorestyle{p}}
         \label{tab:famiglia-5}
\begin{tabular}%
       {llc>{\raggedright\arraybackslash}p{0.21\linewidth}}
\toprule
  Nome     &  Ruolo  & Età & Attività    \\
\midrule
  Giovanni &  padre  & 47  & impiegato   \\
  Maria    &  madre  & 44  & insegnante elementare \\
  Giovanna &  figlia & 14  & studente di~scuola media \\
  Piero    &  figlio &  8  & allievo di~scuola elementare \\
```

TABELLA 3.6 La famiglia Rossi; tabella con colonna p

Nome	Ruolo	Età	Attività
Giovanni	padre	47	impiegato
Maria	madre	44	insegnante elementare
Giovanna	figlia	14	studente di scuola media
Piero	figlio	8	allievo di scuola elementare

```
\bottomrule
\end{tabular}
\end{table}
```

Esempio 3.4 ***Una tabella con note***

Riprendiamo la solita famiglia Rossi e proviamo ad usare le note per specificare, per esempio, il luogo dove ogni membro della famiglia spende il suo tempo lavorativo o scolastico.

Come è già stato rilevato nel capitolo introduttivo, gli ambienti flottanti non accettano le note, né quelle al piede, né quelle marginali. La soluzione a questo problema consiste nello sfruttare l'ambiente *minipage* in modo che questa piccola pagina sia l'oggetto da flottare; tuttavia la piccola pagina è autonoma per quel che riguarda il suo contenuto. Riprendiamo perciò l'intero ambiente *tabular* dell'esempio precedente aggiungendovi le note dentro la paginetta, in modo che tabella e note flottino assieme, figura 3.7.

Il codice usato è il seguente.

```
\begin{table}\centering % mai usare l'ambiente center
%                         dentro gli ambienti flottanti
\caption{La famiglia Rossi;
         tabella con note in calce}
         \label{tab:famiglia-note}
         \begin{minipage}{0.8\linewidth}\centering
\begin{tabular}%
       {llc>{\raggedright\arraybackslash}p{0.4\linewidth}}
```


Tabelle, versione v.1.1.02 del 2021-09-22

Tabella 3.7 La famiglia Rossi; tabella con note in calce

Nome	Ruolo	Età	Attività
Giovanni	padre	47	impiegato[a]
Maria	madre	44	insegnante elementare[b]
Giovanna	figlia	14	studente di scuola media[c]
Piero	figlio	8	allievo di scuola elementare[d]

[a]Presso la ditta Alfa.
[b]Scuola elementare Giovanni Pascoli.
[c]Scuola media Sebastiano Valfrè.
[d]Scuola elementare Giovanni Pascoli.

```
\toprule
Nome     &  Ruolo  & Età & Attività    \\
\midrule
Giovanni &  padre  & 47  & impiegato%
      \footnote{Presso la ditta Alfa.}   \\
Maria    &  madre  & 44  & insegnante elementare%
     \footnote{Scuola elementare Giovanni Pascoli.} \\
Giovanna &  figlia & 14  & studente di~scuola media%
     \footnote{Scuola media Sebastiano Valfrè.} \\
Piero    &  figlio &  8  & allievo di~scuola elementare%
     \footnote{Scuola elementare Giovanni Pascoli.} \\
\bottomrule
\end{tabular}
\let\footnoterule\relax
\end{minipage}
\end{table}
```

Come si vede, le note all'interno di una *minipage* sono etichettate con lettere, invece che con numeri. Il filetto che normalmente precede le note è stato eliminato (rendendone il comando equivalente a `\relax`), visto che la tabella è terminata con il suo filetto scuro. Analogamente, si potrebbe personalizzare il modo di scrivere le note con un piccolo spazio fra il richiamo e il loro

testo; comunque questi sono dettagli che si possono affrontare separatamente, lasciando che questa guida si occupi essenzialmente delle tabelle.

Esempio 3.5 ***Una tabella complessa e piena di filetti***

I filetti vanno sempre usati con estrema parsimonia; la documentazione del pacchetto `booktabs` è esplicita su questo argomento; tuttavia, esistono rari casi in cui è necessario ricorrere ai filetti.

Nella Guida GUIT si parla delle casse tipografiche e ne vengono disegnate un paio; il verbo disegnare, nonostante si parli di tabelle, si spiega perché il disegno di queste casse necessita degli scomparti ben allineati e incolonnati e, quindi, facendone una tabella, bisogna disegnare i filetti.

Si noti che la cassa richiede un centinaio di celle di varie dimensioni, quindi è piuttosto complessa; il codice necessario richiede di comporre la cassa complessiva unendo alcune sottotabelle con dimensioni diverse. Uno dei pregi dell'ambiente *tabular* è che è robusto e quindi può essere annidato in altre tabelle. Per agevolare il compito, si compongono prima alcune tabelle memorizzandole in opportune scatole mediante comandi nativi di TEX, in modo da evitare i problemi che nascono dall'eventuale fragilità di comandi, ma anche per avere una visione più chiara di quale sotto tabella si sta parlando. Si definisce anche una larghezza "standard" delle celle, alcune delle quali potrebbero avere dimensioni multiple della cella di base; in questo modo siamo sicuri che l'intera costruzione rispetti pressapoco la stessa griglia.

Il codice per costruire il disegno della cassa 3.8 è lungo, ma è sufficientemente documentato per spiegare le varie fasi. Non ci si spaventi dell'uso del comando nativo `\def`; in generale è poco prudente farne uso direttamente, perché si corre il rischio di ridefinire comandi interni del kernel o dei pacchetti caricati; ma in questo caso le definizioni sono locali ai vari gruppi, e in particolare a quello formato dall'ambiente *table*. Nello stesso tempo, si possono osservare diversi trucchetti per ottenere in fretta alcuni comandi che alleviano lo sforzo di gestire un numero così elevato di celle nella tabella complessiva.

```
\begin{table}\renewcommand{\arraystretch}{1.8}
\caption{Una cassa tipografica}\label{tab:cassa}
%
% Assegnazione locale del nome \wcella
% al registro dimensionale 10
\dimendef\wcella=10 \wcella=2em
% Alcuni comandi utili
```

```
\def\m #1 #2 {\makebox[0pt]{\llap{#1\enspace}%
  \vrule height1.8\ht\strutbox depth1.8\dp\strutbox
  \rlap{\enspace#2}}}
\def\M #1 {\makebox[0pt]{\parbox{\wcella}%
  {\linespread{.5}\selectfont#1}}}
\def\B #1&{\makebox[\wcella]{#1}&}
\def\BB#1\\{\makebox[\wcella]{#1}\\}

% Tabella delle maiuscole in alto a sinistra
\setbox0\hbox{%
\begin{tabular}{|*7{c|}}\hline
\B A &\B B&\B C   & \B D&\B E&\B F&\BB G         \\\hline
  H  &  I &\m J K &  L  & M  & N  & O            \\\hline
  P  &  Q & R     &  S  & T  & U  & V            \\\hline
  X  &  Y & Z     &  É  & Ê  & Ë  &\M espo-nenti \\\hline
\end{tabular}}

% Tabella delle minuscole accentate e altri segni in alto a destra
\setbox1\hbox{\wcella=1.975em
\begin{tabular}{|*7{c|}}\hline
\B á  &\B é&\B í            &\B ó &\B ú  &\B \ae&\BB \oe \\\hline
 (\,) & §  &\m [\,] $\star$ &\AE  &\OE   &\c C  & \c{c}  \\\hline
 â    & ê  & î              & ô   & û    &\&    & /      \\\hline
 ä    & ë  & ï              & ö   & ü    & k    & j      \\\hline
\end{tabular}}

\wcella=1.285em
\def\B #1&{\makebox[\wcella]{#1}&}%
\def\BB#1\\{\makebox[\wcella]{#1}\\}%
\def\M#1 {\multicolumn2{c|}{#1}}%
\def\MM#1 {\multicolumn2{|c|}{#1}}
\def\R#1 {\multicolumn2{c|}{\raisebox{-10pt}[0pt][0pt]{#1}}}
\def\RR#1 {\multicolumn2{|c|}{\raisebox{0pt}[0pt][0pt]{#1}}}
\def\r#1 {\raisebox{-10pt}[0pt][0pt]{#1}}
\def\b#1 {\makebox[0pt]{\parbox{1.1\wcella}%
                        {\linespread{.5}\selectfont#1}}}
\def\bb#1 {\multicolumn2{c|}{\makebox[0pt]{\parbox{2.7\wcella}%
                                {\linespread{.5}\selectfont#1}}}}
```

```
% Tabella delle minuscole in basso a sinistra
\setbox2\hbox{%
\begin{tabular}{|*9{c|}}\hline
\B à &\B è &\B ì &\B ò &\B ù &\B ---&\B"<\,">&\B fl&\BB ffl\\\hline
ffi  &\M q       &\r b &\r v &\R c           &\R d    \\\cline{1-3}
ff   &\M h       &     &     &\R ~           &\R ~         \\\hline
fi   &\R m       &\R n       &\R o           &\R a    \\\cline{1-1}
z    &\R ~       &\R ~       &\R ~           &\R ~   \\\hline\hline
y    &\R l       &\R t       &\R u           &\R i    \\\cline{1-1}
x    &\R ~       &\R ~       &\R ~           &\R ~         \\\hline
\end{tabular}}

% Tabella delle cifre e di alcuni segni speciali in basso a destra
\setbox3\hbox{%
\begin{tabular}{|*9{c|}}\hline
\MM ~         &\B w &\B W &\B À &\B È&\B Ì&\B Ò&\BB Ù\\\cline{3-9}
\RR e         &\r f &\r g & 1   & 2  &3   &4   &5    \\\cline{5-9}
\MM ~         &     &     & 6   & 7  & 8  &9   &0         \\\hline
\B-   &\B     &\R r &\R s &\b {sp.\ fini} &!   &?    \\
\cline{1-2}                                                 \cline{7-9}
\MM p         &\M ~       &\M ~      &
               \bb{\hbox{\hspace*{.7em}spazi}\hbox{mezzani\strut}}
                                                &.  \\\hline\hline
\RR{\parbox{2\wcella}{\vspace*{15pt}
                      \hbox{\hspace*{.7em}spazi}
                      \hbox{\enspace grossi}
                      \hbox{\footnotesize(terziroli)}}}
             &\R \makebox[0pt]{quadratini}
                           &;    &\r, &\R quadrati
                                                &    \\\cline{5-5}
\M ~         &\M ~        &:    &           &\M ~
                                                 &\rotatebox{90}{%
                                                \makebox[\wcella]{%
                                            \hspace*{1.7em}q.\ tondi}%
}\\\hline
\end{tabular}}

% Assemblaggio delle quattro sotto tabelle
\resizebox{\linewidth}{!}{%
```

Tabella 3.8 Una cassa tipografica

A	B	C	D	E	F	G	á	é	í	ó	ú	æ	œ
H	I	J K	L	M	N	O	( )	§	[ ] ⋆	Æ	Œ	Ç	ç
P	Q	R	S	T	U	V	â	ê	î	ô	û	&	/
X	Y	Z	É	Ê	Ë	espo-nenti	ä	ë	ï	ö	ü	k	j

à	è	ì	ò	ù	—	« »	fl	ffl	e	w	W	À	È	Ì	Ò	Ù
ffi	q		b	v	c		d			f	g	1	2	3	4	5
ff	h											6	7	8	9	0
fi	m		n		o		a		- ,	r		s		sp. fini	!	?
z									p					spazi mezzani		.
y	l		t		u		i		spazi grossi (terziroli)	quadratini		;	,	quadrati		q. tondi
x												:				

```
  \fbox{\tabcolsep=0pt
    \begin{tabular}{r@{\hskip3pt}l}
      \box0          &\box1\\
      \noalign{\vspace{3pt}}
      \box2          &\box3
    \end{tabular}%
  }%
}
\end{table}
```

Esempio 3.6 ***Il pacchetto*** `multirow`

Il pacchetto `multirow` permetterebbe di costruire delle celle che occupano diverse righe in verticale; secondo alcuni, l'uso di simili celle non è un modello di buona tipografia. Sarebbe quindi preferibile non fare uso delle funzionalità di questo pacchetto.

Nell'esempio del precedente paragrafo si sono usate delle celle il cui spazio verticale è adiacente ad alcune righe di celle adiacenti. Ma l'esempio precedente è uno schema che assomiglia ad una tabella, ma ha dei filetti un po' particolari; infatti si è detto che lo si è disegnato ricorrendo a più ambienti *tabular*.

Nello stesso tempo *non* si è usato il comando `\multirow`, definito dal pacchetto *multirow*, perché, nonostante la sua sintassi, che consente di agire con finezza in certe circostanze delicate, è fragile e poco affidabile, nel senso che richiede spesso una delicata messa a punto a seconda di ciò che contiene.

Nello schema della cassa 3.8 si è preferito usare il comando `\parbox` che agisce come l'ambiente *minipage*, ma che raccoglie il suo contenuto mediante il suo terzo argomento, invece che raccogliere tutto quanto fra `\begin{minipage}` e `\end{minipage}`. Il meccanismo con `\parbox`, almeno in questo caso, è sicuramente più facile da gestire.

3.2 L'AMBIENTE *tabular**

Talvolta è necessario costruire delle tabelle che abbiano una larghezza specificata; chi scrive trova che esista anche una qualche giustificazione, se la tabella ha una larghezza naturale di poco superiore alla giustezza del testo circostante e la si vuole restringere un poco senza cambiare il corpo dei font. Si può fare ma lo si descriverà nel prossimo capitolo.

Per allargare una tabella, in modo che impieghi l'intera giustezza del testo, si può usare l'ambiente *tabular** che ha esattamente le stesse caratteristiche di *tabular*, tranne per quel che riguarda la gestione della larghezza.

La sintassi di *tabular** è la stessa di quella di *tabular*, ma richiede un argomento in più.

```
\begin{tabular*}{⟨larghezza⟩}[⟨allineamento⟩]{⟨descrittori delle colonne⟩}
⟨cella⟩ & ⟨cella⟩ & ... & ⟨cella⟩ \\
⟨cella⟩ & ⟨cella⟩ & ... & ⟨cella⟩ \\
...
⟨cella⟩ & ⟨cella⟩ & ... & ⟨cella⟩ \\
\end{tabular*}
```

Per l'⟨*allineamento*⟩ verticale non c'è da aggiungere nulla a quanto già detto per l'ambiente *tabular*. Non c'è da dire molto nemmeno sulla ⟨*larghezza*⟩ della tabella. C'è da aggiungere qualcosa, invece, sui ⟨*descrittori delle colonne*⟩.

Si ricorda che i ⟨*descrittori delle colonne*⟩ non contengono solo i codici che definiscono i tipi di colonne, ma anche i separatori fra le colonne. Per impostazione predefinita, nessuno di quei descrittori è uno spazio elastico:

`\tabcolsep` indica un registro di lunghezza che contiene una lunghezza rigida, non un triplice registro di spazio elastico che, oltre alla lunghezza naturale, contiene anche l'allungamento e l'accorciamento. Quindi non si può giocare su queste grandezze, ma bisogna fare uso di una *@-espressione* che contenga il comando `\extracolsep` specificandogli nell'argomento uno spazio elastico adeguatamente allungabile, tipicamente lo spazio `\fill` che ha una lunghezza naturale nulla, e un allungamento 'moderatamente' infinito.[4]

Esempio 3.7 ***Cosa succede con i filetti verticali***

Come già detto descrivendo questo fatto nel capitolo precedente, l'effetto dello spazio elastico si manifesta solo alla sinistra dell'argomento di ciascuna cella. Una piccola tabella che si voglia stiracchiare fino ad essere larga quanto la giustezza, diventa di una bruttezza indescrivibile se uno non la vede con i propri occhi. Nella tabella 3.9 si vede benissimo quanto descritto qui, pur avendo messo il comando nel primo descrittore di spaziatura a sinistra della prima cella; il fenomeno descritto nasce dal fatto che la direttiva fornita da `\extracolsep` viene letta quando si sta già eseguendo il primo separatore della prima cella, quindi la sua impostazione non può avere luogo quando la spaziatura è già stata eseguita; però si noti come si semplifica la lista dei descrittori se si usa il descrittore `!` messo a disposizione del pacchetto `array`.

Il codice con cui è stato composto l'esempio è il seguente.

```
\begin{table}
\caption[Tre tabelle con lo stesso contenuto]%
{Tre tabelle con lo stesso contenuto:
$(a)$ tabella composta nella sua larghezza naturale;
$(b)$ tabella allargata fino all'intera giustezza;
$(c)$ Tabella allargata ma con il primo separatore
descritto mediante il descrittore  \descrittorestyle{!}.
I filetti verticali evidenziano la causa della cattiva
composizione.}\label{tab:stirata}
```

[4] Gli interpreti del sistema TEX basati su LATEX prevedono tre 'livelli' di infinito sia per l'allungamento sia per l'accorciamento; uno debole, uno medio e uno forte; si riconoscono facilmente quando si incontrano i comandi di spaziatura orizzontale `\hfil` (debole), `\hfill` (medio) e `\hfilll` (forte). Internamente sono rappresentati da codici particolari, perché la macchina di elaborazione non conosce valori di "infinito"; l'utente non deve preoccuparsene, ma è bene che sia al corrente di questa particolarità.

TABELLA 3.9 Tre tabelle con lo stesso contenuto: (*a*) tabella composta nella sua larghezza naturale; (*b*) tabella allargata fino all'intera giustezza; (*c*) Tabella allargata ma con il primo separatore descritto mediante il descrittore `!`. I filetti verticali evidenziano la causa della cattiva composizione.

abc	def	ghi

(*a*)

abc	def	ghi

(*b*)

abc	def	ghi

(*c*)

```
\centering
\begin{tabular}{|*3{l|}}
\hline
abc & def & ghi \\
\hline
\end{tabular}\medskip

\makebox[\linewidth]{$(a)$}\bigskip

\begin{tabular*}{\linewidth}%
    {@{\extracolsep{\fill}\vline\hskip\tabcolsep}%
     *3{l|}%
    }
\hline
abc & def & ghi \\
\hline
\end{tabular*}\medskip

\makebox[\linewidth]{$(b)$}\bigskip

\begin{tabular*}{\linewidth}%
```

Tabella 3.10 Altre due tabelle a confronto: la stessa tabella 3.9 senza filetti verticali (a) composta con *tabular**, o (b) con *widetabular*.

abc	def	ghi

(a)

abc	def	ghi

(b)

```
    {!{\vline\extracolsep{\fill}}*3{l|}}
\hline
abc & def & ghi \\
\hline
\end{tabular*}\medskip

\makebox[\linewidth]{$(c)$}
\end{table}
```

Esempio 3.8 ***Lo stesso esempio senza filetti verticali***

Il codice e l'esempio precedente servono solo per evidenziare il difetto; senza i filetti verticali, il difetto non si vede assolutamente o è del tutto trascurabile, tabella 3.10. Tuttavia, merita confrontare questo risultato con quanto si può ottenere con *widetabular* oppure con *tabularx* (con tutte le colonne di tipo `X`) per rendersi conto della differenza di funzionamento dei diversi metodi e dell'estetica del risultato finale. In questo confronto si tenga presente che il contenuto delle tre celle è molto stretto e la tabella è molto larga e non ha molto senso allargare la tabella fino alla giustezza della gabbia di stampa; l'esempio serve solo per mostrare il diverso funzionamento.

Chi scrive ritiene, comunque, che per comporre tabelle con una larghezza specificata, si ottengano migliori risultati se ci si serve dei pacchetti descritti nel prossimo capitolo.

Esempio 3.9 ***Di nuovo la famiglia Rossi***

Volendo comunque usare l'ambiente *tabular** con la terza colonna che contenga capoversi di diverse parole, bisogna definire una larghezza da passare

al descrittore p; è chiaro che con qualche tentativo si potrebbe determinare la larghezza giusta perché la tabella sia larga quanto la pagina, ma arbitrariamente decidiamo che la larghezza di questa colonna sia pari a `0.5\linewidth`. Si può comporre la tabella con questo codice e usando l'ambiente *tabular**: si ottiene la tabella 3.11.

```
\begin{table}\centering % mai usare l'ambiente center
 %                          dentro gli ambienti flottanti
 \caption{La famiglia Rossi; tabella a giustezza piena}
         \label{tab:famiglia con descrizione}
 \begin{tabular*}{\linewidth}%
        {!{\extracolsep{\fill}}%
         llc>{\raggedright\arraybackslash}p{0.5\linewidth}%
        }
 \toprule
 Nome     &  Ruolo  & Età & \multicolumn1l{Attività} \\
 \midrule
 Giovanni &  padre  & 47  & Impiegato nella ditta Alfa, come
                            responsabile dell'ufficio commerciale.\\
 Maria    &  madre  & 44  & Insegnate di sostegno presso la
                            scuola elementare Giovanni Pascoli.\\
 Giovanna &  figlia & 14  & Studente presso  la scuola secondaria
                            inferiore  Sebastiano Valfrè.\\
 Piero    &  figlio &  8  & Allievo presso la scuola elementare
                            Giovanni Pascoli dove frequenta la
                            classe terza~C. \\
 \bottomrule
 \end{tabular*}
\end{table}
```

Senza filetti verticali il risultato è perfettamente accettabile. Si intende che se il valore arbitrario della giustezza della colonna p fosse stato troppo grande, la compilazione avrebbe prodotto un risultato impresentabile con le righe della quarta colonna sporgenti fuori della tabella.

Questo esempio mostra ancora una volta che, indipendentemente dalla miglior pratica tipografica, cioè dalla *best practice*, e dalle raccomandazioni del pacchetto `booktabs`, i filetti verticali presentano più inconvenienti che vantaggi, tranne in casi eccezionali ed insoliti.

Nel nostro caso si potrebbe pensare di usare più filetti orizzontali per separare maggiormente le informazioni relative a ciascun membro della famiglia;

Tabella 3.11 La famiglia Rossi tabella a giustezza piena

Nome	Ruolo	Età	Attività
Giovanni	padre	47	Impiegato nella ditta Alfa, come responsabile dell'ufficio commerciale.
Maria	madre	44	Insegnate di sostegno presso la scuola elementare Giovanni Pascoli.
Giovanna	figlia	14	Studente presso la scuola secondaria inferiore Sebastiano Valfrè.
Piero	figlio	8	Allievo presso la scuola elementare Giovanni Pascoli dove frequenta la classe terza C.

Tabella 3.12 La famiglia Rossi; tabella larga con filetti

Nome	Ruolo	Età	Attività
Giovanni	padre	47	Impiegato nella ditta Alfa, come responsabile dell'ufficio commerciale.
Maria	madre	44	Insegnate di sostegno presso la scuola elementare Giovanni Pascoli.
Giovanna	figlia	14	Studente presso la scuola secondaria inferiore Sebastiano Valfrè.
Piero	figlio	8	Allievo presso la scuola elementare Giovanni Pascoli dove frequenta la classe terza C.

alternativamente si potrebbero usare delle spaziature maggiori, specificandole come argomento facoltativo del comando `\\`, per esempio, `\\[1ex]`; nelle tabelle 3.12.

Gli ambienti *tabularx* e *widetabular* 4

I pacchetti `tabularx` e `widetable` forniscono all'utente altri due modi per produrre tabelle di larghezza specificata. Il loro funzionamento è concettualmente diverso; infatti:

- il pacchetto `tabularx` raggiunge lo scopo mediante una (o più colonne) di un nuovo tipo, `X`, che assomiglia alla colonna di tipo `p`, ma che si determina da solo la larghezza da assegnare alla colonna; invece

- il pacchetto `widetable` raggiunge lo scopo allargando lo spazio intercolonna, cioè determinando il valore più opportuno del parametro `\tabcolsep` che permette di portare l'intera tabella alla larghezza specificata.[1]

I risultati sono palesemente diversi e la cosa è evidente se sono presenti i filetti verticali; senza filetti verticali, il risultato della composizione con `tabularx` è diverso da quello mostrato nella tabella 3.11, mentre con `widetable` vi assomiglia molto; il vantaggio di usare `widetable` invece dell'ambiente nativo *tabular** è che in presenza di filetti i contenuti delle celle non sono impresentabili come nella tabella 3.9, perché gli spazi sono ugualmente distribuiti a cavallo di ogni separatore di cella.

[1] È opportuno notare che, nelle versioni precedenti alla 2.1 del 2020-10-13, l'ambiente aveva lo stesso nome del pacchetto; con questa versione il nome è stato cambiato in *widetabular*, e il vecchio nome *widetable* è conservato per retrocompatibilità, ma se ne scoraggia l'uso; è meglio riservare il nome "table" e i suoi derivati per l'ambiente flottante, e il nome "tabular" con i suoi derivati per la tabulazione vera e propria.

4.1 Composizione con *tabularx*

Bisogna innanzi tutto caricare il pacchetto `tabularx` nel preambolo del documento; questo pacchetto definisce una nuova colonna `X` ricorrendo alle funzionalità del pacchetto `array`. Questa colonna si allarga o si stringe automaticamente per consentire alla tabella di raggiungere la larghezza desiderata. L'ambiente fa da solo i calcoli necessari per ottenere quella larghezza, ma questi calcoli richiedono di comporre due volte la tabella prima di comporre la tabella finale; tutto ciò avviene dietro le quinte e l'utente non si accorge di niente, se non, forse di un leggerissimo rallentamento nella composizione.

La stessa tabella può contenere diverse colonne di tipo `X`; normalmente sono tutte della stessa larghezza finale; sarebbe possibile renderle "diversamente allungabili", ma la cosa è delicata e bisogna seguire una precisa sequenza di operazioni che vanno viste nella documentazione del pacchetto `tabularx`.

Esempio 4.1 ***Ancora la famiglia Rossi***

Si vuole ricomporre la tabella 3.11 usando l'ambiente *tabularx* invece dell'ambiente *tabular**. Il codice seguente produce il risultato della tabella 4.1.

```
\begin{table}\centering % mai usare l'ambiente center
 %                          dentro gli ambienti flottanti
 \caption{La famiglia Rossi;
          tabella con colonna \descrittorestyle{X}}
          \label{tab:famiglia con tabularx}
 \begin{tabularx}{\linewidth}{llc>{\raggedright\arraybackslash}X}
 \toprule
   Nome     &  Ruolo  & Età & \multicolumn1c{Attività} \\
 \midrule
   Giovanni &  padre  & 47  & Impiegato nella ditta Alfa, come
                              responsabile dell'ufficio commerciale.\\
   Maria    &  madre  & 44  & Insegnate di sostegno presso la
                              scuola elementare Giovanni Pascoli.\\
   Giovanna &  figlia & 14  & Studente presso  la scuola secondaria
                              inferiore  Sebastiano Valfrè.\\
   Piero    &  figlio &  8  & Allievo presso la scuola elementare
                              Giovanni Pascoli dove frequenta la
                              classe terza~C. \\
 \bottomrule
```

Tabella 4.1 La famiglia Rossi; tabella con colonna `X`

Nome	Ruolo	Età	Attività
Giovanni	padre	47	Impiegato nella ditta Alfa, come responsabile dell'ufficio commerciale.
Maria	madre	44	Insegnate di sostegno presso la scuola elementare Giovanni Pascoli.
Giovanna	figlia	14	Studente presso la scuola secondaria inferiore Sebastiano Valfrè.
Piero	figlio	8	Allievo presso la scuola elementare Giovanni Pascoli dove frequenta la classe terza C.

```
 \end{tabularx}
\end{table}
```

Il codice è più semplice e il risultato è più equilibrato. Si noti che, volendo, si potrebbe definire un nuovo tipo di colonna che contenga anche il codice che precede il tipo `X`, Per esempio si potrebbe definire la colonna `Z`, mediante il codice:

```
\newcolumntype{Z}{>{\raggedright\arraybackslash}X}
```

e cambiare la prima riga dell'ambiente *tabularx* in questo modo:

```
\begin{tabularx}{\linewidth}{llcZ}
```

che rende la scrittura del codice ancora più semplice.

4.2 Composizione con *widetabular*

Le cose sono simili con `widetable` e il suo ambiente *widetabular*; se si compila con l'ambiente *tabular** e il risultato non soddisfa, basta sostituire l'indicazione `widetabular` a `tabular*` negli argomenti di `\begin` e `\end`, senza cambiare nient'altro, per ottenere la compilazione della tabella con questo pacchetto. Infatti la sintassi dell'ambiente *widetabular* è semplicemente

```
\begin{widetabular}{⟨larghezza⟩}[⟨posizione⟩]{⟨lista dei descrittori⟩}
...
⟨riga di celle⟩ \\
⟨riga di celle⟩ \\
⟨riga di celle⟩ \\
...
\end{widetabular}
```

La compilazione va a buon fine se e solo se nella ⟨*lista di descrittori*⟩ esiste, esplicitamente o implicitamente, almeno un'istanza della spaziatura ottenuta con `\tabcolsep`. Se invece gli spazi intercolonna sono tutti eseguiti con *@-espressioni* vuote o che non contengono spaziature dell'ammontare espresso mediante il valore di `\tabcolsep`, il pacchetto emette un avviso, ma non modifica gli spazi intercolonna e si comporta come un semplice *tabular*.

Esempio 4.2 ***Di nuovo la famiglia Rossi***

In questo esempio si ricompila la tabella relativa alla famiglia Rossi recuperando il codice che ha portato alla tabella 3.5 e sostituendo il nome dell'ambiente come spiegato sopra:

```
\begin{table}\centering % mai usare l'ambiente center
 %                        dentro gli ambienti flottanti
 \caption{La famiglia Rossi; tabella con spazi allargati}
          \label{tab:famiglia con widetabular}
 \begin{widetabular}{\linewidth}{llcl}
 \toprule
   Nome     &  Ruolo  & Et\`a & Attivit\`a    \\
 \midrule
   Giovanni &  padre  & 47  & impiegato   \\
   Maria    &  madre  & 44  & insegnante elementare \\
   Giovanna &  figlia & 14  & studente di scuola media \\
   Piero    &  figlio &  8  & allievo di scuola elementare \\
 \bottomrule
 \end{widetabular}
 \end{table}
```

Tabella 4.2 La famiglia Rossi; tabella con spazi allargati

Nome	Ruolo	Età	Attività
Giovanni	padre	47	impiegato
Maria	madre	44	insegnante elementare
Giovanna	figlia	14	studente di scuola media
Piero	figlio	8	allievo di scuola elementare

Il risultato è riportato nella tabella 4.2; vale la pena confrontare questa soluzione con quella ottenuta con *tabular*.

Si noti, invece, che se si usa una versione di `widetable` precedente alla 2.0 e se si compila con pdflaex, sarebbe necessario usare i comandi per gli accenti invece delle lettere accentate, come si è esemplificato con il codice precedente. A partire dalla versione 2.0, ciò non è più necessario e si possono tranquillamente usare le lettere accentate anche compilando con pdflatex. Questo problema non si è mai manifestato con xelatex o lualatex nemmeno con le versioni precedenti alla 2.0.

Esempio 4.3 ***Sempre la famiglia Rossi con descrizione delle attività***

Nell'esempio che ha fornito la tabella 3.11, la larghezza della quarta colonna era fissata a `0.5\linewidth`. Gli spazi intercolonna, ottenuti con l'ambiente *tabular**, erano abbastanza grandi tranne quelli a sinistra della prima e a destra della quarta colonna: se si ricompone con *widetabular*, gli spazi intercolonna cambiano un poco, come si vede nella tabella 4.3.

```
\begin{table}\centering % mai usare l'ambiente center
 %                        dentro gli ambienti flottanti
 \caption{La famiglia Rossi;
          tabella con colonna \descrittorestyle{p} e spazi
          allargati}
          \label{tab:famiglia con descrizione e widetabular}
 \begin{widetabular}{\linewidth}%
        {llc>{\raggedright\arraybackslash}p{0.5\linewidth}%
        }
\toprule
   Nome     &Ruolo&Età& \multicolumn1l{Attività} \\
\midrule
```

```
   Giovanni & padre  & 47 & Impiegato nella ditta Alfa, come
                        responsabile dell'ufficio commerciale.\\
   Maria    & madre  &44  & Insegnate di sostegno presso la
                           scuola elementare Giovanni Pascoli.\\
   Giovanna & figlia & 14 & Studente presso la scuola secondaria
                           inferiore Sebastiano Valfr\`e.\\
   Piero    & figlio &  8 & Allievo presso la scuola elementare
                           Giovanni Pascoli dove frequenta
                           la classe terza C.\\
\bottomrule
\end{widetabular}
\end{table}
```

Il confronto di questa tabella 4.3 con quella ottenuta con *tabularx* nella tabella 4.1 mostra delle differenze sostanziali; qui, tabella 4.3 la larghezza della quarta colonna è fissato, là, tabella 4.1, è la quarta colonna che è stata allargata per raggiungere il risultato voluto. Inoltre, è sostanziale la differenza del fatto che con *tabularx* è necessaria almeno una colonna di tipo `X`, mentre con *widetabular* il risultato non dipende dai tipi delle colonne, ma solo dallo spazio intercolonna.

Tabella 4.3 La famiglia Rossi; tabella con colonna `p` e spazi allargati

Nome	Ruolo	Età	Attività
Giovanni	padre	47	Impiegato nella ditta Alfa, come responsabile dell'ufficio commerciale.
Maria	madre	44	Insegnate di sostegno presso la scuola elementare Giovanni Pascoli.
Giovanna	figlia	14	Studente presso la scuola secondaria inferiore Sebastiano Valfrè.
Piero	figlio	8	Allievo presso la scuola elementare Giovanni Pascoli dove frequenta la classe terza C.

Matrici e incolonnamenti matematici 5

Anche in matematica si devono incolonnare numeri, formule, relazioni. Per tabelle di soli numeri si può usare anche l'ambiente *tabular* senza bisogno di ricorrere ad ambienti matematici; tuttavia, come in questa guida, talvolta si impostano le cifre minuscole che in matematica e nelle tabelle non hanno un bell'aspetto; per questo, in matematica LaTeX impiega solo le cifre maiuscole, perciò negli incolonnamenti, che contengono colonne di numeri, col kernel di LaTeX si usa l'ambiente *array*.

D'altra parte, in matematica esistono molti altri tipi di incolonnamenti; fra i pacchetti disponibili nel sistema TeX c'è il pacchetto `amsmath` che contiene le definizioni di diversi altri incolonnamenti per produrre matrici variamente delimitate; incolonnamenti di espressioni matematiche, specialmente quelle che formano sistemi; insiemi di condizioni. Quel pacchetto è talmente importante che andrebbe sempre caricato per comporre qualsiasi documento che contenga matematica di livello appena superiore a quello delle scuole elementari.

Bisogna però tenere presente un aspetto importante: le espressioni matematiche di solito non vengono composte in linea col testo, a meno che non si tratti di espressioni semplicissime. Quindi ogni espressione matematica fuori testo occupa non poco spazio verticale, non meno dell'equivalente di tre righe. Lo spazio prima della formula dipende da quanto questa sia larga e da quanto sia corto il precedente *righino*, cioè la riga di testo di lunghezza minore della giustezza che precede la formula. Anche lo spazio dopo può essere minore. Non si scende nei dettagli, perché in fondo sono irrilevanti in questa guida. L'importante è che bisogna essere ben consci che le espressioni matematiche non possono essere rese flottanti perché fanno parte del flusso del discorso. A maggior ragione sono oggetti ingombranti in senso verticale se sono formati da matrici o da incolonnamenti particolari

che non possono essere divisi fra più pagine; un caso evidente si ha quando un incolonnamento è affiancato da opportuni delimitatori. Per esempio, le soluzioni dell'equazione di secondo grado:

$$x^2 + 2ax + b = 0$$

sono date da:

$$x_{1,2} = \begin{cases} -a \pm \sqrt{a^2 - b} & \text{se} \quad a^2 > b \\ -a & \text{se} \quad a^2 = b \\ -a \pm \mathrm{i}\sqrt{b - a^2} & \text{se} \quad a^2 < b \end{cases}$$

Il problema si presenta anche con i sistemi di equazioni, visto che in Italia, spesso, tali sistemi sono raccolti in unico blocco mediante una graffa:

$$\begin{cases} 3x + 4y - 2z = 15 \\ 2x - 5y + z = 11 \\ x + 2y + 4z = 2 \end{cases}$$

Oppure nelle equazioni matriciali come la seguente:

$$\begin{pmatrix} 3 & 4 & -2 \\ 2 & -5 & 1 \\ 1 & 2 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 15 \\ 11 \\ 2 \end{pmatrix}$$

In questi casi non è possibile spezzare l'espressione, quindi è possibile che per riempire la pagina vengano dilatati gli spazi attorno ad altri oggetti fuori testo, o almeno gli spazi elastici che precedono ogni capoverso nella pagina.

O si cambia il testo, oppure si crea una pagina mozza, oppure si allunga o si accorcia un capoverso senza cambiargli il testo ma richiedendo al programma, mediante il comando `\looseness`, di comporlo con una riga in più o una in meno, oppure si usa il comando `\enlargethispage` (con o senza asterisco), il cui uso, però, non è banale; è difficile elencare tutti i casi possibili, e l'utente deve decidere caso per caso quale sia la soluzione migliore; può essergli utile esaminare i commenti della guida tematica *Il LaTeX Reference Manual commentato* reperibile nella sezione documentazione del forum del GuIT.

In altre circostanze, gli allineamenti non si spezzano per impostazione predefinita, ma è possibile autorizzare quest'operazione o per il singolo allineamento o per tutti gli allineamenti del documento; il pacchetto `amsmath` mette a disposizione il comando `\displaybreak[`⟨*numero*⟩`]` per autorizzare un fine pagina esattamente nel punto in cui lo si inserisce (ovviamente solo in adiacenza al comando `\\` all'interno di un singolo incolonnamento; oppure il comando `\allowdisplaybreaks[`⟨*numero*⟩`]` per autorizzare l'operazione in tutti gli incolonnamenti del documento. Con il ⟨*numero*⟩, che può essere un intero nell'intervallo chiuso $[0, 4]$ (come per il comando `\pagebreak[`⟨*numero*⟩`]` da usare in modo testo) si regola 'la forza' del comando.

Il kernel di LaTeX definisce solo due ambienti per eseguire degli incolonnamenti matematici; il pacchetto `amsmath` ne definisce una moltitudine. I due ambienti del kernel sono *array* e *eqnarray*. Quelli di `amsmath` sono: *aligned*, *multline*, *multline**, *split*, *gather*, *gather**, *align*, *aligned**, *alignat*, *falign*, *falign**, *cases*, *matrix*, *pmatrix*, *bmatrix*, *Bmatrix*, *vmatrix*, *Vmatrix*.

Alcuni allineamenti vanno usati dentro gli ambienti *equation* o *equation**, che impostano l'ambente matematico in cui essi operano. Gli ambienti asteriscati non numerano le singole righe, mentre gli altri, tranne quelli per le matrici, numerano progressivamente ogni equazione; esiste, però, l'ambiente *subequations*, che permette di numerare un sistema con un unico numero e di numerare le singole righe del sistema (non racchiuse con la graffa) con una sottonumerazione letterale.

I sistemi del kernel e quelli di `amsmath` sono leggermente diversi; anche la loro sintassi è leggermente diversa, ma è un dettaglio secondario; la differente composizione si nota nelle equazioni (5.1) e (5.2).

$$\begin{aligned} x + 4y - 2z &= 15 \\ l2x - 5y + z &= 11 \\ x + 2y + 4z &= 2 \end{aligned} \tag{5.1}$$

$$\begin{aligned} 3x + 4y - 2z &= 15 \\ 2x - 5y + z &= 11 \\ x + 2y + 4z &= 2 \end{aligned} \tag{5.2}$$

Come si osserva chiaramente, le spaziature attorno al segno = sono diverse; nel sistema (5.1) quegli spazi sono più grandi di quelli che compaiono nel sistema (5.2); gli spazi più larghi sono errori, ma sono un residuo di retrocompatibilità con il vecchio LaTeX 209, defunto nel 1994, ma conservati per consentire di compilare ancora documenti 'd'annata'. Si eviti pertanto di usare l'ambiente *eqnarray*, ma si usi solo l'ambiente *align* al suo posto.

Gli ambienti *array* del kernel e *matrix* di `amsmath` producono risultati quasi equivalenti, difficili da riconoscere ad occhio nudo. Tuttavia, quando costituiscono il cuore di una matrice delimitata, i risultati diventano più visibili: i delimitatori dei numerosi ambienti per matrici di `amsmath` sono più accostati alle colonne esterne delle matrici. Nella guida tematica, già citata, sulla composizione della matematica ci sono molti esempi di matrici che vengono composte meglio con l'ambiente del kernel, ma sono matrici tanto particolari che vanno studiati caso per caso e non costituiscono la regola, per cui nella stragrande maggioranza dei casi gli ambienti di `amsmath` sono sempre da preferire rispetto a quelli del kernel.

Nelle pagine precedenti si sono già mostrati alcuni esempi; ricordo solo che *matrix* produce una matrice non delimitata; *pmatrix* produce una matrice delimitata da parentesi tonde; *bmatrix* produce matrici delimitate da parentesi quadre; con *Bmatrix* i delimitatori sono le parentesi graffe; con *vmatrix* si producono determinanti, cioè la loro matrice è racchiusa fra barre verticali; infine *Vmatrix* produce le norme, cioè le loro matrici sono delimitate con doppie barre verticali; si veda la tabella 5.1.

Esempio 5.1 ***I sei tipi di matrici del pacchetto*** `amsmath`

Vale la pena di mostrare l'aspetto grafico dei sei tipi di matrici che si possono comporre con gli ambienti definiti da `amsmath`. Essi sono riportati nell'ordine nella tabella 5.1, dove ognuna è stata composta replicando lo stesso schema con il solo cambiamento del nome dell'ambiente:

```
\begin{minipage}[c][0.15\textwidth][c]{0.15\textwidth}
\[
\begin{{⟨matrix⟩}}
1 & 2 & 3\\
4 & 5 & 6\\
7 & 8 & 9
\end{{⟨matrix⟩}}
```

Tabella 5.1 I sei tipi di matrici di amsmath

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

(*matrix*)

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

(*pmatrix*)

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

(*bmatrix*)

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

(*Bmatrix*)

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

(*vmatrix*)

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

(*Vmatrix*)

```
\]
\end{minipage}
```

A questo scopo è stato definito il comando `\tipomatrice` in questo modo:

```
\newcommand\tipomatrice[1]{%
  \begin{minipage}[c][0.20\textwidth]{0.20\textwidth}
    \[
      \begin{#1}
      1 & 2 & 3\\
      4 & 5 & 6\\
      7 & 8 & 9
      \end{#1}
    \]\vss
    \centering(\amb{#1})%
  \end{minipage}%
}
```

Il codice per costruire l'intera tabella 5.1 diventa allora semplicemente il seguente.

```
\begin{table}
\caption{I sei tipi di matrici di \packstyle{amsmath}}
\label{tab:matrici}
```

```
\centering
\begin{tabular}{*3c}
\tipomatrice{matrix}
   & \tipomatrice{pmatrix}
      & \tipomatrice{bmatrix}
\\~\\
\tipomatrice{Bmatrix}
   & \tipomatrice{vmatrix}
      & \tipomatrice{Vmatrix}
\end{tabular}
\end{table}
```

Esempio 5.2 ***Spezzare una lunga equazione con*** *multline*

Gli ambienti di allineamento, invece, sono particolari: *multline* spezza una lunga equazione in più monconi e mette il numero dell'equazione alla fine dell'ultimo moncone, sulla stessa linea o, se dovesse interferire, subito sotto; il primo moncone è allineato a sinistra, con un certo margine, l'ultimo è allineato a destra con un margine adeguato; gli eventuali altri monconi sono centrati. Chi scrive non ha mai dovuto ricorrere a questo ambiente, e per presentare questo esempio ha faticato a trovare una vera espressione matematica adatta; tuttavia, matematici e fisici incontrano queste situazioni molto spesso.

L'equazione spezzata (5.3):

$$\begin{aligned}\mathrm{Ker}_n(x) = -[\log(x/2)+\gamma] + \tfrac{1}{4}\pi\,\mathrm{Bei}_n(x)\\ + \frac{1}{2}\sum_{k=0}^{n-1}\frac{(n-k-1)!(x/2)^{2k-n}}{k!}\cos\frac{(3n+2k)\pi}{4}\\ + \frac{1}{2}\sum_{k=0}^{\infty}\frac{(x/2)^{n+2k}}{k!(n+k)!}[\varPhi(k)+\varPhi(n+k)]\cos\frac{(3n+2k)\pi}{4}\end{aligned} \tag{5.3}$$

è stata composta con il codice seguente:

```
\begin{multline}
  \Ker_n(x)  = -[\log(x/2)+\upgamma]
                      +{\textstyle\frac{1}{4}}\pi\Bei_n(x)\\
+\frac{1}{2}\sum_{k=0}^{n-1}
             \frac{(n-k-1)!(x/2)^{2k-n}}{k!}
```

```
                                    \cos\frac{(3n+2k)\pi}{4}\\
+\frac{1}{2}\sum_{k=0}^\infty
     \frac{(x/2)^{n+2k}}{k!(n+k)!}[\Phi(k)+\Phi(n+k)]
                              \cos\frac{(3n+2k)\pi}{4}
\label{equ:Ker}
\end{multline}
```

Esempio 5.3 ***Formule allineate e spezzate su più righe***

La formula (5.4) contiene un incolonnamento con il secondo membro spezzato su più righe; dove e quando spezzare una formula? La regola dettata dal matematico Knuth, il padre del sistema TEX, colui il quale ha creato questo sistema perché non era soddisfatto della perdita di professionalità dei tecnici addetti alla composizione della matematica con i mezzi messi a disposizione dagli elaboratori disponibili alla fine degli anni settanta del secolo scorso[1] è la seguente: si va a capo dopo un operatore binario e non lo si ripete all'inizio della riga successiva; non ci sono problemi con gli operatori di somma, sottrazione, divisione, e molti altri; solo l'operatore di moltiplicazione va reso visibile, visto che solitamente lo si sottintende; lo si rende visibile con un punto centrato sull'asse matematico se il secondo termine è una variabile, oppure con il segno $\times$ se è un valore numerico. Non guasta se i monconi di formula che seguono il primo sono rientrati rispetto al primo.

L'equazione (5.3) viene ripetuta usando l'ambiente *aligned*:(5.4).

$$\begin{aligned}\mathrm{Ker}_n(x) = &-[\log(x/2)+\gamma] + \tfrac{1}{4}\pi\,\mathrm{Bei}_n(x)\\ &+\frac{1}{2}\sum_{k=0}^{n-1}\frac{(n-k-1)!(x/2)^{2k-n}}{k!}\cos\frac{(3n+2k)\pi}{4}\\ &+\frac{1}{2}\sum_{k=0}^{\infty}\frac{(x/2)^{n+2k}}{k!(n+k)!}[\Phi(k)+\Phi(n+k)]\cos\frac{(3n+2k)\pi}{4}\end{aligned} \tag{5.4}$$

Il codice usato è il seguente:

```
\begin{equation}
\begin{aligned}
```

[1] La professionalità tipografica di |aTeX per la composizione della matematica è ancora insuperata, mentre la professionalità dei tecnici che non usano il sistema TEX non è assolutamente migliorata

```
  \Ker_n(x)  &=
     -[\log(x/2)+\upgamma]
            +{\textstyle\frac{1}{4}}\uppi\Bei_n(x)\\
    &\qquad+\frac{1}{2}\sum_{k=0}^{n-1}
      \frac{(n-k-1)!(x/2)^{2k-n}}{k!}
            \cos\frac{(3n+2k)\uppi}{4}\\
    &\qquad+\frac{1}{2}\sum_{k=0}^\infty
     \frac{(x/2)^{n+2k}}{k!(n+k)!}[\Phi(k)+\Phi(n+k)]
            \cos\frac{(3n+2k)\uppi}{4}
\end{aligned}
\label{equ:aligned}
\end{equation}
```

Le piccole differenze che si notano fra la formula (5.3) e (5.4) è la posizione del numero della formula e l'incolonnamento in bandiera delle due ultime espressioni.

Se si fosse composto con l'ambiente *split*, non si sarebbero notate differenze di sorta rispetto alla composizione con *aligned*; in realtà visivamente non ce ne sono, ma in background si vede una notevole differenza; infatti componendo con *aligned*, tutta la formula rimane all'interno di un rettangolo circoscritto (*bounding box*) che è tangente alla parte scritta: invece con *split* il rettangolo circoscritto arriva fino al margine destro della griglia di stampa. Chi scrive preferisce usare sempre *aligned*.

Esempio 5.4 ***Incolonnamento senza allineamento***

L'ambiente *gather* serve per incolonnare delle espressioni matematiche prescindendo dagli operatori che contengono, ma semplicemente centrandole nello spazio disponibile. Se, per esempio, si prendono le equazioni di Maxwell relative al campo elettromagnetico (nella forma conforme alle norme ISO, che usano il SI, e vietano l'uso dei vari sistemi CGS), le si può incolonnare come nel sistema (5.5)

$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \tag{5.5a}$$

$$\nabla \times \boldsymbol{H} = \frac{\partial \boldsymbol{D}}{\partial t} + \boldsymbol{J} \tag{5.5b}$$

$$\nabla \cdot \boldsymbol{D} = \varrho \tag{5.5c}$$
$$\nabla \cdot \boldsymbol{B} = 0 \tag{5.5d}$$
$$\boldsymbol{B} = \boldsymbol{\mu}\boldsymbol{H} \tag{5.5e}$$
$$\boldsymbol{D} = \boldsymbol{\varepsilon}\boldsymbol{E} \tag{5.5f}$$

Per gli ingegneri che sono abituati a lavorare con materiali anisotropi la permeabilità $\boldsymbol{\mu}$ e la permettività $\boldsymbol{\varepsilon}$ sono tensori; le altre grandezze in corsivo nero sono vettori. Tutta la simbologia rispetta le norme ISO.

Il codice usato per comporre il sistema (5.5) è il seguente:

```
\begin{subequations}\label{equ:Maxwell-1}
\def\arraystretch{1.5}%
\begin{gather}
\nabla \times \mathbfit{E} =
      - \frac{\partial \mathbfit{B}}{\partial t}\\
\nabla \times \mathbfit{H} =
        \frac{\partial \mathbfit{D}}{\partial t}
           +\mathbfit{J}\\
\nabla\cdot\mathbfit{D} = \rho\\
\nabla\cdot\mathbfit{B} =0\\
 \mathbfit{B} = \mathsfbfit{\mu}\mathbfit{H}\\
 \mathbfit{D} = \mathsfbfit{\epsilon}\mathbfit{E}
\end{gather}
\end{subequations}
```

I comandi usati per le variabili in neretto sono tutti definiti nel pacchetto `pm-isomath`, di cui si consiglia l'uso se e solo se si compone con pdflatex; con lualatex e xelatex si carica il pacchetto `unicode-math` e gli si specifica l'opzione *math-style=ISO*; i comandi per gestire la scelta dei font nelle formule sono identici o molto simili a quelli indicati in questi esempi.

Esempio 5.5 ***Incolonnamento con allineamento***

Se invece dell'ambiente *gather* si usa l'ambiente *align*, si può ottenere

l'incolonnamento con i vari termini allineati sulla base dei segni =.

$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \tag{5.6a}$$
$$\nabla \times \boldsymbol{H} = \frac{\partial \boldsymbol{D}}{\partial t} + \boldsymbol{J} \tag{5.6b}$$
$$\nabla \cdot \boldsymbol{D} = \varrho \tag{5.6c}$$
$$\nabla \cdot \boldsymbol{B} = 0 \tag{5.6d}$$
$$\boldsymbol{B} = \boldsymbol{\mu H} \tag{5.6e}$$
$$\boldsymbol{D} = \boldsymbol{\varepsilon E} \tag{5.6f}$$

Il codice usato per comporre la nuova versione delle equazioni di Maxwell, (5.6), ben incolonnate e allineate, è il seguente.

```
\begin{subequations}\label{equ:Maxwell-2}
\def\arraystretch{1.5}%
\begin{align}
\nabla \times \mathbfit{E} =
      - \frac{\partial \mathbfit{B}}{\partial t}\\
\nabla \times \mathbfit{H} =
        \frac{\partial \mathbfit{D}}{\partial t}
           +\mathbfit{J}\\
\nabla\cdot\mathbfit{D} = \rho\\
\nabla\cdot\mathbfit{B} =0\\
 \mathbfit{B} = \mathsfbfit{\mu}\mathbfit{H}\\
 \mathbfit{D} = \mathsfbfit{\epsilon}\mathbfit{E}
\end{align}
\end{subequations}
```

Va da sé che questa rapida scorsa degli ambienti e dei comandi forniti dal pacchetto `amsmath` è riduttiva; esiste la sua ampia documentazione oltre alla guida tematica per la composizione della matematica a cui fare riferimento. Tuttavia, anche gli incolonnamenti e gli allineamenti matematici si riferiscono alle stesse idee che sovrintendono alle tabelle. Nel kernel di LaTeX sono sostanzialmente gestiti dallo stesso codice interno, con l'unica differenza che le celle delle tabelle *array* sono predisposte per ricevere materiale matematico, mentre quelle di *tabular* lo sono per ricevere

materiale testuale. In entrambi i tipi di tabelle si può commutare da un tipo all'altro, ma bisogna esplicitarlo o cella per cella, o colonna per colonna; non è comodo, quindi i due ambienti hanno la loro ragion d'essere.

Le tabelle sporgenti nel margine 6

Nei capitoli precedenti si è parlato di tabelle come di oggetti monolitici, che quando sono un po' troppo larghi vanno gestiti in vari modi: ridotti, con spazi intercolonna minori (*widetabular* ci riesce, a meno che la larghezza specificata sia inferiore alla larghezza naturale della tabella), scalandole con `\resizebox`, riducendo il corpo dei font con cui sono composte, eccetera. Ma un modo particolare è quello che si ottiene facendole protrudere nel margine esterno della pagina. Una idea è quella che possano protrudere nel margine esterno quanto basta per arrivare ad occupare anche lo spazio destinato alle note marginali, quindi larghe quanto vale la somma `\textwidth + \marginparsep + \marginparwidth`.

Ci sono diverse strade per ottenere il risultato. Il codice che Leslie Lamport ha adottato per creare i suoi esempi, che appunto protrudono nel margine esterno, ricorre al meccanismo delle note marginali; in sostanza crea un "brano di testo" formato da un pilastrino alto quanto la figura o la tabella; poi vi mette accanto una nota marginale, che da sola va nel margine esterno; ma per ottenere il risultato desiderato deve mettere la figura o la tabella dentro una scatola `\makebox` che ne nasconda la vera larghezza. Insomma, il codice è abbastanza complesso e, a conoscenza di chi scrive, non lo si trova già disponibile in pacchetti preconfezionati.

Esempio 6.1 ***Una tabella che sporge nel margine***

Per questa documentazione si è definito un ambiente *widebox* che costruisce eventualmente una scatola di larghezza pari a `\textwidth + \marginparsep + \marginparewidth` contenente un oggetto (in questo caso una tabella) e la inscatola in una `\makebox` a cui si specifica la giustezza del testo pari a `\textwidth`, ma si specifica l'opzione di inserimento nella scatola con `l` o `r` ricevuto come argomento facoltativo dall'apertura dell'ambiente *widebox*. Non

Tabella 6.1 Tabella che sporge nel margine esterno

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32
3	3	6	9	12	15	18	21	24	27	20	33	36	39	42	45	48
4	4	8	12	16	20	24	28	32	36	20	44	48	52	56	60	64
5	5	10	15	20	25	30	35	40	45	20	55	60	65	70	75	80
6	6	12	18	24	30	36	42	48	54	20	66	72	78	84	90	96
7	7	14	21	28	35	42	49	56	63	20	77	84	91	98	105	112
8	8	16	24	32	40	48	56	64	72	20	88	96	104	112	120	128
9	9	18	27	36	45	54	63	72	81	20	90	108	117	126	135	144
10	10	20	30	40	50	60	70	80	90	20	110	120	130	140	150	160
11	11	22	33	44	55	66	77	88	99	20	121	132	143	154	165	176
12	12	24	36	48	60	72	84	96	108	20	132	144	156	168	180	192
13	13	26	39	52	65	78	91	104	117	20	143	156	169	182	195	208
14	14	28	42	56	70	84	98	112	126	20	154	168	182	196	210	224
15	15	30	45	60	75	90	105	120	135	20	165	180	195	210	225	240
16	16	32	48	64	80	96	112	128	144	20	176	192	208	224	240	256

c'è nessun automatismo; tocca all'utente di specificare il codice di posizione che preferisce oppure di non specificare niente. Se non specifica nessun codice di posizione, l'ambiente colloca il suo contenuto centrato nella pagina e, quindi sporge in entrambi i margini; altrimenti, specificando `r` il contenuto viene posto nella scatola accostato a destra, e quindi sporge nel margine sinistro; viceversa, se si specifica `l`, il contenuto è allineato a sinistra e sporge nel margine destro.

Benché manchi ogni automatismo, l'uso di questo ambiente è semplice, perché basta aggiustare il codice di posizione durante la revisione delle bozze finali. La grande tabella 6.1 è stata messa in posizione scegliendo a posteriori il codice di posizione adatto; inizialmente era stata composta senza specificare niente e, pur sporgendo in entrambi i margini, non era sgradevole; tuttavia, dopo la prima compilazione, con la quale la tabella appariva in una pagina di una data parità, è stato scelto il codice di posizione adatto per farla sporgere solo nel margine esterno.

In questa documentazione ,che è composta con i margini interno ed esterno uguali, l'allineamento destro o sinistro non è molto importante, specialmente se

viene letta a schermo; quindi anche la posizione centrata andrebbe benissimo. Tuttavia se si stampa questo documento e se ne raccolgono le pagine in un raccoglitore, oppure se lo si confeziona in brossura incollata, quindi con il dorso abbastanza rigido, è probabilmente meglio scegliere l'allineamento con la protrusione solo nel margine esterno anche quando si dispone di margini uguali.

L'ambiente *widebox* può essere usato anche fuori dall'ambiente flottante, anche se non è consigliabile, quando l'oggetto è una tabella; però, di solito, le tabelle sono ingombranti anche verticalmente, quindi è meglio lasciarle flottare affinché LaTeX stesso trovi il posto migliore per comporne il contenuto; come si vede nella tabella 6.1, la didascalia è composta prima dell'ambiente *widebox*, in modo che risulta collocata come per tutte le altre tabelle. Volendo, si potrebbe mettere il comando `\caption` con i suoi argomenti dentro l'ambiente *widebox* e la didascalia apparirebbe centrata sulla tabella.

Il codice, da inserire completamente nel preambolo del documento, è il seguente.

```
\usepackage{xparse}
\newbox{\Wbox}

\NewDocumentEnvironment{widebox}{ O{c} }{%
\linewidth=\dimexpr \textwidth + \marginparsep
    + \marginparwidth\relax
\begin{lrbox}{\Wbox}%
}{\end{lrbox}%
\noindent
\ifdim\wd\Wbox < \linewidth
  \makebox[\textwidth][c]{\box\Wbox}%
\else
  \makebox[\textwidth][#1]{\box\Wbox}%
\fi}
```

Si vuole far notare che è stato usato il comando `\NewDocumentEnvironment`, messo a disposizione dal pacchetto `xparse`, in quanto l'argomento facoltativo viene usato solo dai comandi di chiusura. Con il comando del kernel di LaTeX, `\newenvironment`, ciò non sarebbe possibile, se non conservandone il valore in macro locali. Usando `xparse` e le sue funzionalità, il tutto è molto più semplice.

Si noti, infine, che qualsiasi ambiente per creare tabelle descritto nei capitoli precedenti può comparire dentro l'ambiente *widebox*; nel caso della tabella 6.1 si è usato l'ambiente *widetabular* specificandogli la solita larghezza `\linewidth` che all'interno del nuovo ambiente è definita apposta per comprendere oltre alla giustezza del testo anche lo spazio orizzontale occupato dalle note marginali.

Le tabelle ruotate 7

7.1 I problemi che nascono nel ruotare le tabelle

Talvolta una tabella è più larga che alta e non rimane dentro la giustezza della gabbia, ma ne esce di più di quanto non si riesca ad aggiustare facendola sporgere nel margine.

Una soluzione accettabile consiste nel ruotarla insieme alla sua didascalia; volendo, si potrebbe ruotare solo la tabella e lasciare la didascalia normalmente sopra o sotto la tabella ruotata, ma non sarebbe conforme alla buona pratica tipografica.

Un punto dolente è che esistono pacchetti per ruotare le tabelle o altri oggetti, e per impostazione predefinita le ruotano in modo da lasciare sempre la didascalia dal lato del margine esterno. Questo obbligherebbe il lettore a inclinare il capo verso sinistra per leggere una tabella ruotata che si trovi su una pagina di destra e, viceversa, dovrebbe inclinare la testa dall'altro lato quando la tabella è su una pagina di sinistra. Peggio ancora, se una tabella lunga è divisa fra due pagine affacciate; la prima metà è sulla pagina di sinistra e la seconda metà sulla pagina di destra e, per leggere la tabella complessiva, il lettore dovrebbe inclinare il capo ora da un lato ora dall'altro. Assolutamente inammissibile. Se poi il testo che sta leggendo è stampato, il lettore dovrebbe essere particolarmente veloce nel controruotare il libro o il fascicolo ora a destra ora a sinistra a seconda di quale pagina stia leggendo.

La soluzione consiste nel ruotare la tabella di 90° sempre in senso antiorario, indipendentemente dal fatto che la tabella cada su una pagina pari o dispari.

7.2 La parità della pagina

Un altro problema nasce per le lunghe tabelle larghe che vengono suddivise fra più pagine; bisogna sempre cominciare la prima sezione su una pagina pari (di sinistra) quando si compone (e si stampa) fronte-retro; invece non ha importanza la parità della pagina dove cade la prima parte della tabella quando si compone solamente fronte.

Il kernel di LaTeX contiene il comando `\cleardoublepage` che chiude la pagina corrente e, se la pagina successiva è pari, inserisce una pagina "vuota" e poi ricomincia a comporre sulla successiva pagina dispari. Questo comando è usato dagli stessi comandi `\part` e `\chapter` per le classi *book* e *report* e classi affini; non è importante per la classe *article* dove il semplice comando `\clearpage` passa direttamente alla pagina successiva senza preoccuparsi della parità. La classe *memoir* dispone di comandi che permettono di controllare il numero della pagina corrente e quindi anche di differire certe parti della composizione fino a quando la pagina successiva ha la parità desiderata.

Analogamente, il comando `\afterpage` differisce la composizione del suo argomento solo dopo la fine della pagina corrente, e indipendentemente dalla parità della nuova pagina.

Se la classe con cui si sta lavorando non dispone di nessun mezzo per differire certe parti di composizione a pagine con una data parità, in particolare a pagine pari, bisogna definirsi comandi appositi per svolgere questo compito.

Non è difficile, perché basta inserire nel preambolo del proprio documento, usando `\usepackage`, o in un file di macro personali, usando `\RequirePackage`, un file che contenga il poco codice seguente.

```
% Commentare e decommentare a seconda di dove si mette il codice

%\usepackage{afterpage}    % per il preambolo del documento
%\makeatletter             % per il preambolo del documento

\RequirePackage{afterpage} % per un file .sty personale
\newcommand\supaginapari[1]{%
  \afterpage{%
    \ifodd\value{page}\expandafter\@firstoftwo
    \else\expandafter\@secondoftwo
```


Tabelle, versione v.1.1.02 del 2021-09-22

```
    \fi{\supaginapari{#1}}{#1}
  }
}
\newcommand\supaginadispari[1]{%
  \afterpage{%
    \ifodd\value{page}\expandafter\@secondoftwo
    \else\expandafter\@firstoftwo
    \fi{\supaginadispari{#1}}{#1}
  }
}
```

Entrambi i comandi `\supaginapari` e `\supaginadispari` controllano la parità della pagina disponibile dopo l'esecuzione del primo `\afterpage`; se la parità corrisponde a quella indicata nel nome del comando, inseriscono l'argomento nel buffer d'entrata del motore di composizione; altrimenti invocano lo stesso comando con lo stesso argomento; questo verrà eseguito quando la pagina con la parità sbagliata sarà completata. Un po' arzigogolata? Forse, ma funziona; non per niente è un'idea che scaturisce da una comunicazione privata con David Carlisle, l'autore del pacchetto `afterpage`.

Si notino i comandi `\expandafter` per invertire l'ordine di espansione dei comandi; e l'uso delle due macro `\@`firstoftwo e `\@`secondoftwo che vengono mandati in esecuzione dopo che il test `\ifodd` è stato completato, così da renderlo robusto rispetto al contenuto dell'argomento indicato con `#1`.

Come David Carlisle ha sottolineato, il comando `\afterpage` interagisce con l'algoritmo con il quale il motore di composizione accoda ogni pagina composta al file di uscita; quindi può avere effetti indesiderati se il comando `\afterpage` cade troppo vicino ad altri comandi (come quelli di sezionamento) che agiscono anche loro sullo stesso algoritmo. Di solito non è un problema, ma quando accade, si può notare che il contenuto dell'argomento di `\afterpage` non è stato composto affatto, apparentemente perso per la strada, oppure quell'argomento viene composto ma interagisce malamente con i comandi di sezionamento. Non è difficile, caso per caso, trovare una soluzione a questo problema. In effetti, prima di escludere il pacchetto `tabu` da questa guida, si era cercato di comporre la lunga tabella con la nomenclatura con l'ambiente *longtabu*; si era impostato l'inizio su una pagina pari; poiché il codice era verso la fine del capitolo, l'ordine

è stato eseguito, ma ha impedito di cominciare il capitolo successivo su una nuova pagina. Si è inserito il comando `\newpage` ma la lunga tabella cadeva dentro il capitolo successivo e dentro pagine in cui le testatine già si riferivano la nuovo capitolo. La semplice soluzione è consistita nello spostare all'indietro tutto il codice relativo alla tabella, mettendolo subito all'inizio dell'esempio in cui veniva discusso il codice; questo era bastato per risolvere l'interazione critica fra `\afterpage` e il comando di sezionamento `\chapter`. In altri casi si possono presentare problemi da risolvere diversamente, ma, di solito, spostare il codice sorgente risolve la situazione.

Si può usare `\afterpage` anche per ritardare la composizione di una lunga tabella ruotata. Per quel che riguarda la rotazione, esistono i pacchetti `lscape` e `pdflscape`, che funzionano anche con tabelle che si svolgono su più pagine, che però, ruotano la tabella, o la successione delle sottotabelle, che si svolgono su più pagine, ma nel file finale da leggere a schermo, controruotano anche la pagina visibile sullo schermo cosicché, quando si legge, non c'è bisogno di inclinare la testa. Si rimanda alla documentazione dei due pacchetti per i vari dettagli e le impostazioni consigliabili.

Recentemente è apparso il nuovo pacchetto `hvfloat` che permette di ruotare di angoli qualsiasi e in modo indipendente sia il contenuto del suo argomento (tabella o figura) sia la sua didascalia; il lettore è invitato a leggerne la documentazione, perché le possibilità di gestione sono diverse, compresa quella di non far flottare il contenuto, cioè di eseguire qualcosa di simile al risultato del comando di posizione `H`, già descritto in connessione con il pacchetto `float`, e di cui si è raccomandato di non fare uso se non in rari casi del tutto eccezionali. Tra le altre cose il pacchetto `hvfloat` permette di comporre le didascalie anche accanto all'oggetto fatto flottare, invece che sopra o sotto.

Esempio 7.1 ***Una tabella ruotata***

Chi scrive ha ottenuto risultati ottimi con un semplice nuovo ambiente che ruota una tabella con la sua didascalia solo su una pagina, lasciando la testatina e il piedino nella posizione abituale. Il nuovo ambiente è definito con il codice seguente:

```
\newenvironment{ruotascatola}{%
\begin{lrbox}{0}\vtop\bgroup\hsize=1.3\textwidth
\textwidth=\hsize \linewidth=\hsize \columnwidth=\hsize
```


Tabelle, versione v.1.1.02 del 2021-09-22

Tabella 7.1 Unità di misura legalmente ammesse in Italia

Grandezza fisica	Unità	Simbolo	Equivalenza
angolo piano	grado sessagesimale	°	$1° = \pi\mathrm{rad}/180$
angolo piano	minuto sessagesimale	′	$1' = \pi\mathrm{rad}/10\,800$
angolo piano	secondo sessagesimale	″	$1'' = \pi\mathrm{rad}/648\,000$
angolo piano	gon o grado centesimale	gon	$1\mathrm{gon} = \pi\mathrm{rad}/200$
angolo piano	giro	giro	$1\mathrm{giro} = 2\pi\mathrm{rad}$
area	ara	a	$1\mathrm{a} = 100\mathrm{m}^2$
area	ettaro	ha	$1\mathrm{ha} = 10\,000\mathrm{m}^2$
volume	litro	l, L	$1\mathrm{L} = 1\mathrm{dm}^3$
tempo	minuto	min	$1\mathrm{min} = 60\mathrm{s}$
tempo	ora	h	$1\mathrm{h} = 3600\mathrm{s}$
tempo	giorno	d	$1\mathrm{d} = 86\,400\mathrm{s}$
massa	tonnellata	t	$1\mathrm{t} = 1000\mathrm{kg}$
massa	unità di massa atomica	u	$1\mathrm{u} = 1,660\,57 \times 10^{-27}\mathrm{kg}$
lavoro, energia	elettronvolt	eV	$1\mathrm{eV} = 1,602\,19 \times 10^{-19}\mathrm{J}$
lavoro, energia	kilowattora	kW h	$1\mathrm{kWh} = 3,6\mathrm{MJ}$
carica elettrica	amperora	A h	$1\mathrm{Ah} = 3600\mathrm{C}$
temperatura Celsius	grado Celsius	°C	$1°\mathrm{C} = 1\mathrm{K}$ ma differisce lo zero della scala: $t = T - 273,15\mathrm{K}$

```
\parindent=0sp\centering
}{\egroup\end{lrbox}\rotatebox{90}{\box0}}
```

Usando questo nuovo ambiente, la cui definizione si riduce veramente a poche righe di codice, diventa semplice ruotare una intera tabella con la sua didascalia di 90° e farla flottare in una pagina di soli oggetti flottanti. Ecco il codice per costruire e ruotare la tabella 7.1 ottenuta rivisitando una tabella presente nella citata guida tematica sulla composizione della matematica delle grandezze.

```
\begin{table}[p]\centering
\begin{ruotascatola}\centering
\caption{Unità di misura legalmente ammesse in Italia}
\label{tab:unita ammesse}
\begin{tabular}{llcl}
\toprule
\multicolumn{1}{c}{Grandezza fisica}
        &\multicolumn{1}{c}{Unità}
                &\multicolumn{1}{c}{Simbolo}
                        &\multicolumn{1}{c}{Equivalenza} \\
\midrule
angolo piano
  & grado sessagesimale
    & ${}^{\circ}$
      & $1^\circ=\uppi\unit{rad}/180$\\
% ...                                          righe omesse
temperatura Celsius
  & grado Celsius
    & ${}^\circ{\mathrm{C}}$
        & $1\unit{^\circ C}=1\unit{K}$ \\
  &  &  & ma differisce lo zero della scala:\\
  &  &  & $t = T - 273,15\unit{K}$ \\[.5ex]
\bottomrule
\end{tabular}
\end{ruotascatola}
\end{table}
```

Si osservi che il comando `\centering` all'interno dell'ambiente *table* serve per centrare orizzontalmente la tabella, mentre lo stesso comando `\centering` dentro l'ambiente *ruotascatola* serve per centrare la tabella verticalmente.

Gli ambienti *longtable* e *supertabular* 8

Gli ambienti descritti nei paragrafi precedenti costruiscono tabelle inscatolate; cioè inserite dentro registri-scatola che vengono gestiti come oggetti monolitici; in particolare, nessuna tabella costruita con quegli ambienti può svilupparsi a cavallo di un fine pagina. Questo è in palese contrasto con quanto succede componendo tabelle con i comuni word processor.

Tuttavia, ha senso disporre di ambienti che costruiscano (lunghe) tabelle che si sviluppano su più pagine; gli esempi di tali tabelle sono innumerevoli, ma quello più ovvio è il caso di un catalogo dove le informazioni vanno bene distinte: prodotto, codice, fotografia, caratteristiche salienti, costo, eccetera. Ovviamente, le informazioni delle varie colonne devono essere composte con le stesse caratteristiche grafiche da una pagina all'altra, in particolare le colonne devono mantenere la stessa larghezza; e da una pagina all'altra devono essere ripetute le intestazioni delle colonne; eventualmente a fondo pagina potrebbe esserci una annotazione che la tabella prosegue e a inizio pagina potrebbe esserci una annotazione che si tratta di una continuazione dalla pagina precedente; non sarebbe male se in ogni pagina fosse ripetuta la didascalia che appare all'inizio della lunga tabella.

Fra i pacchetti del sistema TEX, `longtable` e `supertabular` permettono di comporre simili tabelle; non sono gli unici ma sono quelli usati più frequentemente. Nelle varie guide generali e tematiche che si trovano nella sezione Documentazione del forum del GUIT, ci sono diversi esempi di tali lunghe tabelle.

Chi scrive ha avuto modo di usare le funzionalità di entrambi i suddetti pacchetti, poi ha preferito usare sempre `longtable` anche se, sotto certi aspetti, `supertabular` potrebbe essere più performante. La differenza dipende essenzialmente dal modo in cui viene calcolata la larghezza delle colonne e come questa venga conservata da una pagina all'altra.

È opportuno ricordare che i pacchetti `longtable` e `supertabular` sono compatibili con molti altri pacchetti che agiscono sull'aspetto e la forma delle celle, per esempio con `booktabs` e `array`. Sono invece incompatibili con pacchetti che eseguano raggruppamenti verticali di celle, specificatamente con il diffusissimo pacchetto `multirow`;che in verità sarebbe raccomandabile di evitare, perché non produce tabelle professionali, ma serve solo per imitare le tabelle composte con i *word processor*; se ne è già parlato nel paragrafo 3.6. Ma qui il divieto di usare `multirow` è assoluto, perché interferisce con il meccanismo di `longtable` e `supertabular` per dividere al lunga tabella nei vari monconi da comporre in pagine successive.

Ovviamente qui non è il caso di scendere nei dettagli, ma si spiega come fare nei due casi.

8.1 Comporre una lunga tabella con *longtable*

Innanzi tutto l'ambiente *longtable* non è flottate. Quindi, prima di comporlo nel documento finale, conviene comporlo in un documento a parte e importarne il codice all'inizio di una nuova pagina. Conviene procedere con un semplice documento di servizio, per esempio il seguente:

```
% Documento con un nome di comodo
\documentclass[⟨opzioni⟩]{⟨classe⟩}
...
\usepackage{longtable}
...
\begin{document}
\input{⟨TabellaLunga⟩}
\end{document}
```

Il preambolo di questo documento serve solo per gestire il file che contiene il codice per la lunga tabella, ma è meglio che abbia lo stesso preambolo del documento vero, quello per il quale si sta predisponendo la lunga tabella. Quando questa tabella sarà pronta, allora il suo file verrà importato nel file del documento finale; questo file di servizio potrà servire per altre tabelle lunghe semplicemente cambiando il nome del file da importare ⟨*TabellaLunga*⟩.

A fianco di questo documento, si predispone il file, senza nessun preambolo, con lo stesso nome ⟨*TabellaLunga*⟩ scelto per la messa a punto

necessaria. Perciò questo file che contiene solo il codice avrà una forma del genere:

```
% Inizio file
\begin{longtable}[⟨posizione orizzontale⟩]{⟨descrittori delle colonne⟩}
⟨righe per configurare l'intestazione della prima pagina⟩
\endfirsthead
⟨righe per configurare le intestazioni delle pagine successive⟩
\endhead
⟨righe per configurare il piede dell'ultima pagina⟩
\endlastfoot
⟨righe per configurare il piede delle pagine precedenti⟩
\endfoot
⟨righe della tabella⟩
\end{longtable}
% Fine file
\endinput % Facoltativo ma utile
```

A differenza dell'ambiente *tabular* e dei suoi consimili, il codice di ⟨*posizione orizzontale*⟩ indica solo se tutta la tabella lunga, che si svolge su più pagine, sia da comporre accostata al margine sinistro della griglia di stampa, o al margine destro, oppure sia centrata; i codici di posizione accettati, pertanto, sono solo `l`, `r` e `c` (default).

I ⟨*descrittori delle colonne*⟩ sono gli stessi che si usano per *tabular*.

Nel testo precedente, "prima pagina", " pagine successive", "pagine precedenti" e "ultima pagina" si riferiscono ai vari monconi della tabella, che si trovano su un certo numero di pagine consecutive, e i "piedi" non sono i piedini delle pagine ma le ultime righe dei vari monconi di tabella.

La didascalia va inserita con il solito comando `\caption` come prima riga del primo moncone di tabella, quindi nella prima delle ⟨*righe per configurare l'intestazione della prima pagina*⟩. Nelle intestazioni delle pagine successive si può ripetere la didascalia, ma con il comando `\caption*`, un comando specifico di *longtable* che, normalmente, non è disponibile per tutte le classi di documento, ma solo in alcune, come, per esempio, *memoir*. Il comando `\caption`, specifico per l'ambiente *longtable*, accetta il solito argomento facoltativo e quello obbligatorio, come avviene per tutti gli altri comandi `\caption` degli ambienti flottanti; invece `\caption*` accetta solo

il testo della didascalia, ma non agisce né sulla numerazione delle tabelle, né sull'indice delle tabelle. Il comando `\label` per attribuire un'etichetta alla tabella lunga, a cui fare riferimento con i soliti comandi `\ref` e `\pageref`, va messo solo dopo l'argomento obbligatorio di `\caption`, evidentemente solo fra i comandi inseriti nelle ⟨*righe per configurare le intestazioni della prima pagina*⟩.

Il troncamento in monconi avviene solo dopo i comandi di fine riga tabellare, quindi è bene che l'ultima riga di ogni moncone, tranne l'ultimo, sia composta in modo tale che si capisca che la tabella non è terminata.

Quando la composizione dell'intera tabella lunga è a posto, nel file del documento vero si importa con `\input` solo il file ⟨*TabellaLunga*⟩ nell'argomento del comando `\afterpage`:

```
% Documento da comporre
\documentclass[⟨opzioni⟩]{⟨classe⟩}
...
\usepackage{longtable}
\usepackage{afterpage}
...
\begin{document}
...
\afterpage{\input{⟨TabellaLunga⟩}}

⟨spazio verticale⟩
  % riga vuota
...
\end{document}
```

L'artificio di usare il pacchetto `afterpage` e il suo comando `\afterpage` permette di essere sicuri che la composizione della lunga tabella sia differita all'inizio di una nuova pagina. Lo spazio verticale serve per distanziare la fine della tabella da quanto segue sull'ultima pagina della tabella; secondo le circostanze, potrebbe essere utile inserire un comando `\newpage` per ricominciare la composizione su una nuova pagina.

Esempio 8.1 ***L'inizio di una lunga tabella composta con*** *longtable*

In questa guida sarebbe fuori luogo comporre una tabella lunga per davvero solo a titolo di esempio, comunque ci si riferirà a parte della lunga tabella

che forma la nomenclatura scientifica, riportata nella guida tematica *Regole e consigli per comporre la matematica delle scienze sperimentali*, tabella 8.1. Vale la pena però di riportare le righe di impostazione delle intestazioni e dei piedi di tabella.

```
\afterpage{%
\setlength{\LTleft}{0pt}\setlength{\LTright}{0pt}
\begin{longtable}{%
          @{\extracolsep{0pt plus 1fil minus 1fil}}
          >{\hstrut}p{0.6\textwidth}<{\vspace*{1ex}}
          >{$}c<{$}
          l
          @{}
          }
%
  \caption{Nomenclatura, simboli e unità di misura
           (composta con \ambstyle{longtable})}
           \label{tab:nomenclatura con longtable}\\
  \toprule
  \multicolumn1c{\textbf{Grandezza}}&
    \multicolumn1c{\textbf{Simbolo}}&
    \multicolumn1c{\textbf{Unità SI}}\\
    \midrule
\endfirsthead
%
  \multicolumn3l{\emph{Continua tabella \thetable}}\\
  \midrule
    \multicolumn1c{\textbf{Grandezza}}&
    \multicolumn1c{\textbf{Simbolo}}&
    \multicolumn1c{\textbf{Unità SI}}\\
    \midrule
\endhead
%
  \midrule
  \multicolumn3r{\emph{continua}}
\endfoot
%
```

```
  \bottomrule
\endlastfoot
%
angolo piano        &\alpha,\beta,\gamma,\dots& rad \\
angolo solido       &\omega,\,{\Omega} & sr \\
...
coppia              & T,\,M            & N\,m, (N\,m/rad) \\
\end{longtable}}
```

Non ci si preoccupi se in queste righe compaiono comandi ignoti, come `\hstrut`; si tratta solo di un pilastrino verticale invisibile di altezza specificata per distanziare meglio gli ascendenti delle righe dai filetti `\hline` che li precedono; questo pilastrino era usato nella tabella originale, ma in questo esempio non serve, perché si sono usati i filetti orizzontali del pacchetto `boktabs`.

Si noti, invece, che sono stati usati diversi codici del pacchetto `array` per configurare le colonne. L'argomento di `\extracolsep` non è il solito spazio elastico, ma uno spazio di larghezza naturale nulla, dotato, però, non solo di allungamento, come `\fill`, ma anche di *accorciamento* infiniti (del primo ordine) che aiutano anche a rimpicciolire la larghezza della tabella in modo che resti nella giustezza della gabbia del testo.

Come si vede, al piede del primo moncone della tabella appare all'estrema destra la parola *continua*, in modo che il lettore sappia che la tabella prosegue nella pagina successiva. Del resto, anche l'intestazione del secondo moncone contiene all'estrema sinistra l'indicazione che si sta continuando la tabella; queste indicazioni di solito sono molto utili nelle lunghe tabelle in cui le voci della prima colonna non sono ordinate alfabeticamente.

Le singole intestazioni fanno largo uso di comandi `\multicolumn` per rendere le intestazioni più evidenti rispetto al contenuto delle varie righe della tabella. Si tenga però presente che, quando le celle multicolonna si svolgono su più di una colonna, è possibile che occorrano diverse compilazioni del documento affinché le larghezze delle colonne nei vari monconi diventino uniformi da una pagina alla successiva. La documentazione di `longtable` spiega perché, e indica dei modi per diminuire, se non annullare, questa necessità di compilazioni ripetute. Non è un grande problema, perché, comunque, quando si compone un documento complesso, è difficile che esso non abbia bisogno di aggiustamenti, correzioni di refusi, e simili modifiche, che richiedono in ogni coso diverse ricompilazioni.

Si noti infine che, dopo la tabella, il testo ordinario prosegue dopo uno spazio di una riga vuota; anche se non è stato aggiunto nessun comando di spaziatura verticale, lo spazio preimpostato sembra adeguato.

Ultima osservazione: la tabella comincia all'inizio di una pagina pari perché è stata messa in posizione con il comando `\supaginapari` di cui si è parlato nel paragrafo 7.2.

8.2 Comporre con *supertabular*

La documentazione di `longtable` contiene l'ultimo aggiornamento datato 2014; la documentazione si `supertabular` risale invece al 2004. Non si devono trarre conclusioni affrettate; ci sono altri pacchetti che hanno subito l'ultimo aggiornamento negli anni novanta, ma sono vivi e vegeti ancora oggi.

Il pacchetto offre quattro diversi ambienti per comporre lunghe tabelle.

supertabular funziona come *tabular*, ma spezza il suo contenuto su diverse pagine.

*supertabular** funziona come *tabular** ma spezza il suo contenuto su diverse pagine.

mpsupertabular funziona come *supertabular*, ma ogni moncone di tabella è racchiuso dentro una *minipage* in modo che se in quel moncone compaiono delle note, queste vengono composte in calce al moncone, e non al piede dell'ultima pagina della lunga tabella.

*mpsupertabular** funziona come *supertabular**, ma gestisce le note come *mpsupertabular*.

L'impostazione delle intestazioni e dei piè di tabella, avviene inserendo le righe di configurazione come argomento di appositi comandi.

`\tablefirsthead{`⟨*righe per configurare le intestazioni del primo moncone*⟩`}`
In queste righe si mettono indicazioni simili a quelle indicate per l'ambiente *longtable* tranne la didascalia, per la quale esistono appositi comandi descritti qui sotto.

`\tablehead{`⟨*righe per configurare le intestazioni dei monconi successivi*⟩`}`
Queste righe contengono informazioni simili a quelle indicate per *longtable*.

Tabella 8.1 Nomenclatura, simboli e unità di misura, (composta con *longtable*)

Grandezza	**Simbolo**	**Unità SI**
angolo piano	$\alpha, \beta, \gamma, \ldots$	rad
angolo solido	ω, Ω	sr
lunghezza	l	m
larghezza	b	m
altezza	h	m
raggio	r	m
spessore	d, δ	m
diametro	d	m
percorso curvilineo	s	m
superficie, area	S, A	m^2
volume	V, v	m^3
lunghezza d'onda	λ	m, (m/onda)
numero d'onda ($1/\lambda$)	σ	m^{-1}, (onde/m)
ondulanza ($2\pi/\lambda$)	k	m^{-1}
attenuazione spaziale	α	m^{-1}, (Np/m)
costante di fase	β	m^{-1}
costante di propagazione ($\alpha + i\beta$)	γ	m^{-1}
tempo	t	s
periodo	T	s, (s/ciclo)
frequenza	f	Hz, (cicli/s)

continua

Continua tabella 8.1

Grandezza	**Simbolo**	**Unità SI**
pulsazione	ω	s^{-1}, (rad/s)
tempo di rilassamento o costante di tempo	τ	s, (s/Np)
coefficiente di smorzamento	δ	s^{-1}, (Np/s)
decremento logaritmico (T/τ)	Λ	(Np/ciclo)
velocità	v, u	m/s
velocità angolare	ω	rad/s
accelerazione	a	$\mathrm{m/s^2}$
accelerazione angolare	α	$\mathrm{rad/s^2}$
accelerazione di gravità	g	$\mathrm{m/s^2}$
costante di gravitazione	G	$\mathrm{N\,m^2/kg^2}$
velocità della luce nel vuoto	c_0	m/s
massa	m	kg
massa volumica	ϱ	$\mathrm{kg/m^3}$
densità relativa (all'acqua)	d	–
volume massico ($1/\varrho$)	v	$\mathrm{m^3/kg}$
quantità di moto	p	kg m/s
momento della quantità di moto	L	$\mathrm{kg\,m^2/s}$
momento quadratico di superficie	I	$\mathrm{m^4}$
momento di inerzia	J	$\mathrm{kg\,m^2}$
forza	F	N
coppia	T, M	N m, (N m/rad)

`\tabletail{`⟨*righe per configurare il piede di ogni moncone tranne l'ultimo*⟩`}`

Tabelle, versione v.1.1.02 del 2021-09-22

Queste righe contengono informazioni simili a quelle indicate per *longtable*.

`\tablelasttail{`⟨*righe per configurare il piede dell'ultimo moncone*⟩`}` .

Per la didascalia esistono tre comandi specifici che si comportano come il normale comando`\caption`, ma collocano la didascalia in posti diversi.

`\topcaption[`⟨*didascalia breve*⟩`]{`⟨*didascalia*⟩`}` mette la ⟨*didascalia*⟩ prima della tabella.

`\bottomcaption[`⟨*didascalia breve*⟩`]{`⟨*didascalia*⟩`}` mette la ⟨*didascalia*⟩ dopo la tabella.

`\tablecaption[`⟨*didascalia breve*⟩`]{`⟨*didascalia*⟩`}` mette la ⟨*didascalia*⟩ nella posizione predefinita che di solito è prima della tabella.

Il funzionamento di *supertabular* e dei suoi consimili per spezzare la lunga tabella in monconi avviene dopo ogni comando `\\` controllando ad ogni sua istanza se nella pagina c'è ancora sufficiente spazio; se non c'è abbastanza spazio, chiude il moncone inserendovi l'impostazione per il piede, esegue un fine pagina, poi riapre l'ambiente per un nuovo moncone inserendovi le intestazioni opportune e prosegue fino al completamento del contenuto dell'intero ambiente *supertabular*.

La documentazione non indica come faccia a mantenere costante la larghezza delle colonne nei vari monconi. Segnala, invece, che è possibile che l'ultimo moncone non contenga nessuna riga della tabella, ma solo l'intestazione e il piede e suggerisce come fare per evitare questa situazione (sostanzialmente allargando l'intera tabella con un valore di `\arraystretch` maggiore di 1).

Come si è spiegato sopra, chi scrive non usa più questo ambiente da anni e non ha abbastanza esperienza per indicarne i pregi e i difetti a confronto di quelli di *longtable*. Questo non vuol dire che le sue preferenze personali indichino l'opportunità di non usare *supertabular*; ognuno segua le sue preferenze personali, dopo aver sperimentato con entrambi gli ambienti.

Esempio 8.2 ***L'inizio di una lunga tabella composta con*** *supertabular*

Qui si ripresenta come esempio di lunga tabella quella stessa usata nell'esempio 8.1 ricompilata con *supertabular*.

L'inizio del codice è il seguente.

```
\begin{center}
  \tablecaption{Nomenclatura, simboli e unità di misura;
                con \amb{supertabular}}%
                \label{tab:nomenclatura con supertabular}\\
\tablefirsthead{%
    \hline
    \multicolumn1c{\hstrut\textbf{Grandezza}}&
    \multicolumn1c{\textbf{Simbolo}}&
    \multicolumn1c{\textbf{Unità SI}}\\
    \hline}
    %
\tablehead{%
    \multicolumn3l{\emph{Continua tabella \thetable}}\\
    \noalign{\smallskip\hrule\medskip}
    \multicolumn1c{\textbf{Grandezza}}&
    \multicolumn1c{\textbf{Simbolo}}&
    \multicolumn1c{\textbf{Unità SI}}\\
    \hline}
%
\tabletail{%
    \hline
    \multicolumn3r{\hstrut\emph{continua}}\\}
%
\tablelasttail{%
    \hline}
%
\begin{supertabular}{@{}
    >{\hstrut}p{.6\textwidth}<{\vspace*{1ex}}
    >{$}c<{$}
    l
    @{}
    }
angolo piano  &\alpha,\beta,\gamma,\dots & rad \\
angolo solido &\omega,\,{\Omega}         & sr  \\
...
coppia        & T,\,M                    & N\,m, (N\,m/rad)\\
\end{supertabular}
```

```
\end{center}
```

In questo caso, le lunghe tabelle 8.2 e 8.1 sono quasi identiche, quindi non sembra esserci nessuna differenza fra i pro e i contro per l'uno o l'altro ambiente. In realtà ci sono alcune differenze.

Appare diverso, invece, il modo di gestire il codice della lunga tabella. Innanzi tutto, le personalizzazioni delle intestazioni e dei piedi dei monconi della tabella sono diversi; persino il codice per la didascalia appare prima di `\begin{supertabular}`. Inoltre, quei comandi per contenere le intestazioni e i piedi delle tabelle parziali sembrano essere comandi fragili, al punto di non poter contenere certi altri comandi di uso comune. Il codice dell'inizio della tabella appare, quindi, un pochino diverso da quello delle corrispondenti righe di personalizzazione mostrato nell'esempio 8.1.

Inoltre, e questa è la differenza maggiore, sia i comandi di personalizzazione sia la tabella vera e propria vanno contenute dentro un gruppo o un ambiente; qui si è scelto l'ambiente *center*, anche se la tabella occupa da sola tutta la giustezza della gabbia del testo, ma la presenza di questo ambiente assicura da solo un minimo di spaziatura fra il testo che precede e quello che segue la tabella. Con un'altra tabella si sarebbero potuti usare sia *flushleft* sia *flushright* con risultati analoghi di distanziamento fra il testo precedente e quello seguente. Non volendo usare questi ambienti, si sarebbe potuto inserire tutto il codice fra `\begingroup` e `\endgroup`, ma si sarebbe dovuto provvedere esplicitamente a inserire spaziature verticali, prima e dopo la tabella.

Tabella 8.2 Nomenclatura, simboli e unità di misura; con *supertabular*

Grandezza	**Simbolo**	**Unità SI**
angolo piano	$\alpha, \beta, \gamma, \ldots$	rad
angolo solido	ω, Ω	sr
lunghezza	l	m
larghezza	b	m

continua

Continua tabella 8.2

Grandezza	Simbolo	Unità SI
altezza	h	m
raggio	r	m
spessore	d, δ	m
diametro	d	m
percorso curvilineo	s	m
superficie, area	S, A	m^2
volume	V, v	m^3
lunghezza d'onda	λ	m, (m/onda)
numero d'onda ($1/\lambda$)	σ	m^{-1}, (onde/m)
ondulanza ($2\pi/\lambda$)	k	m^{-1}
attenuazione spaziale	α	m^{-1}, (Np/m)
costante di fase	β	m^{-1}
costante di propagazione ($\alpha + i\beta$)	γ	m^{-1}
tempo	t	s
periodo	T	s, (s/ciclo)
frequenza	f	Hz, (cicli/s)
pulsazione	ω	s^{-1}, (rad/s)
tempo di rilassamento o costante di tempo	τ	s, (s/Np)
coefficiente di smorzamento	δ	s^{-1}, (Np/s)
decremento logaritmico (T/τ)	Λ	(Np/ciclo)

continua

Tabelle, versione v.1.1.02 del 2021-09-22

Continua tabella 8.2

Grandezza	**Simbolo**	**Unità SI**
velocità	v, u	m/s
velocità angolare	ω	rad/s
accelerazione	a	$\mathrm{m/s^2}$
accelerazione angolare	α	$\mathrm{rad/s^2}$
accelerazione di gravità	g	$\mathrm{m/s^2}$
costante di gravitazione	G	$\mathrm{N\,m^2/kg^2}$
velocità della luce nel vuoto	c_0	m/s
massa	m	kg
massa volumica	ϱ	$\mathrm{kg/m^3}$
densità relativa (all'acqua)	d	–
volume massico ($1/\varrho$)	v	$\mathrm{m^3/kg}$
quantità di moto	p	kg m/s
momento della quantità di moto	L	$\mathrm{kg\,m^2/s}$
momento quadratico di superficie	I	$\mathrm{m^4}$
momento di inerzia	J	$\mathrm{kg\,m^2}$
forza	F	N
coppia	T, M	N m, (N m/rad)

In conclusione, i due esempi presentati producono risultati quasi identici; sta dunque all'utente vedere se trova più semplice configurare le intestazioni e i piedi dei monconi di tabella con le funzionalità e i limiti di `longtable` o di `supertabular`.

8.3 Una lunga tabella con incolonnamenti particolari

Nel Forum del GUIT un frequentatore ha richiesto suggerimenti in modo da poter comporre una lunga tabella contenente dati letterali e numerici di vario genere; le colonne con i dati numerici dovevano avere i dati incolonnati in base al separatore decimale; a destra di alcuni valori andava messo ad esponente fino a tre asterischi, che non doveva modificare l'incolonnamento del valore numerico. Enrico Gregorio ha suggerito una soluzione eccellente che ricorre al pacchetto `siunitx`, con il quale *longtable* è perfettamente compatibile. Per ottenere il risultato voluto, con gli eventuali asterischi ad apice, è reso mediante il comando `\rob` che racchiude l'esponente in una scatola di larghezza nulla, cosicché non modifica gli allineamenti; i valori numerici sono composti in una colonna di tipo `S` adeguatamente configurata; per i dettagli dei parametri di configurazione il lettore è invitato a riferirsi alla documentazione del pacchetto `siunitx` adeguatamente aggiornato.

Gli spazi fra le colonne sono troppo larghi, ma il frequentatore del forum del GUIT aveva chiesto espressamente che la tabella occupasse l'intera larghezza della griglia di stampa; il codice qui non è stato modificato per niente, ma la giustezza di questo testo non è la stessa del testo a cui si riferiva quel frequentatore, quindi la tabella lunga è centrata nella giustezza, ma non esattamente uguale alla giustezza. Se si volesse avare una lunga tabella espandibile ad una larghezza specificata, allora sarebbe meglio usare l'ambiente *tabu* (vedi il capitolo 9) che offre questa funzionalità.

Dependent Variable	(1)	(2)
Dep var	XXX	XXXXX
VAR1	0.426^{***}	0.976^{*}
	(0.896)	(0.043)
VAR2	-0.852^{***}	-0.067^{***}
	(0.301)	(0.018)
VAR3	0.029	0.152
	(0.102)	(0.200)
VAR4	0.489^{***}	0.965^{***}
	(0.032)	(0.411)

Dependent Variable	(1)	(2)
Dep var	XXX	XXXXX
VAR5	0.521	0.059
	(0.787)	(0.428)
VAR6	0.178***	0.168***
	(0.062)	(0.038)
VAR7	0.003***	0.117***
	(0.022)	(0.023)
VAR8	0.064***	0.066***
	(0.024)	(0.024)
VAR9	0.132***	0.133***
	(0.026)	(0.024)
VAR10	−0.012	−0.016
	(0.014)	(0.038)
VAR11	−0.015	−0.012
	(0.038)	(0.015)
VAR12	0.059**	0.059**
	(0.102)	(0.018)
VAR13	Yes	Yes
VAR14	Yes	Yes
VAR	9.071***	9.340***
	(0.150)	(0.375)
VAR	1.988	1.988
VAR	0.201	0.925
tests		
tests		
variables		
Robust score		
Regression base		
Test		

Tabelle, versione v.1.1.02 del 2021-09-22

Dependent Variable	(1)	(2)
Dep var	XXX	XXXXX
Ho=		
Critical value		
threshold		
t=		
Over		
Ho=No		
test		
test		

Robust standard errors in parentheses: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$

Il codice con cui è stata composta quella tabella è il seguente. si potrebbero ancora aggiungere le istruzioni per le righe iniziali dei monconi di tabella eccetto il primo, e le ultime righe di ogni moncone tranne l'ultimo, ma non si è voluto espressamente modificare quanto Enrico Gregorio ha suggerito nel forum. Per questo motivo, non si sono aggiunti nemmeno la didascalia e il comando `\label` per disporre dell'indicazione della tabella anche nell'apposito elenco e per potervisi riferire con i soliti comandi `\ref` e `\pageref`.

```
\newcommand{\rob}[1]{\makebox[0pt][l]{\textsuperscript{#1}}}

\begin{longtable}{
  l
  *{2}{
    @{\hspace{6em}}
    S[table-format=-1.4,
      table-space-text-pre={(},
      table-space-text-post={)},
      table-align-text-post=false,
      input-symbols=()
    ]
  }
  @{\hspace{1em}}
```

Tabelle, versione v.1.1.02 del 2021-09-22

```
}
\toprule
Dependent Variable  &  {(1)}           & {(2)}            \\
\cmidrule(lr){2-3}
Dep var             &  {XXX}           & {XXXXX}          \\
\midrule
\endhead
\multicolumn{3}{@{}l@{}}{\footnotesize
  Robust standard errors in parentheses: *** $p<0.01$,
    ** $p<0.05$, * $p<0.1$}
\endlastfoot
%
VAR1                & 0.426\rob{***}  & 0.976\rob{*}     \\
                    & (0.896)         & (0.043)          \\
\addlinespace
VAR2                & -0.852\rob{***} & -0.067\rob{***}  \\
                    & (0.301)         & (0.018)          \\
\addlinespace
VAR3                & 0.029           & 0.152            \\
                    & (0.102)         & (0.200)          \\
\addlinespace
VAR4                & 0.489\rob{***}  & 0.965\rob{***}   \\
                    & (0.032)         & (0.411)          \\
\addlinespace
VAR5                & 0.521           & 0.059            \\
                    & (0.787)         & (0.428)          \\
\addlinespace
VAR6                & 0.178\rob{***}  & 0.168\rob{***}   \\
                    & (0.062)         & (0.038)          \\
\addlinespace
VAR7                & 0.003\rob{***}  & 0.117\rob{***}   \\
                    & (0.022)         & (0.023)          \\
\addlinespace
VAR8                & 0.064\rob{***}  & 0.066\rob{***}   \\
                    & (0.024)         & (0.024)          \\
\addlinespace
VAR9                & 0.132\rob{***}  & 0.133\rob{***}   \\
```

```
                    & (0.026)          & (0.024)          \\
\addlinespace
VAR10               & -0.012           & -0.016           \\
                    & (0.014)          & (0.038)          \\
\addlinespace
VAR11               & -0.015           & -0.012           \\
                    & (0.038)          & (0.015)          \\
\addlinespace
VAR12               &  0.059\rob{**}   &  0.059\rob{**}   \\
                    & (0.102)          & (0.018)          \\
\addlinespace
VAR13               & {Yes}            & {Yes}            \\
\addlinespace
VAR14               & {Yes}            & {Yes}            \\
\addlinespace
VAR                 &  9.071\rob{***}  &  9.340\rob{***}  \\
                    & (0.150)          & (0.375)          \\
\addlinespace
VAR                 &  1,988           &  1,988           \\
\addlinespace
VAR                 &  0.201           &  0.925           \\
\addlinespace
tests               &                  &                  \\
tests               &                  &                  \\
variables           &                  &                  \\
Robust score        &                  &                  \\
Regression base     &                  &                  \\
Test                &                  &                  \\
H0=                 &                  &                  \\
Critical value      &                  &                  \\
threshold           &                  &                  \\
t=                  &                  &                  \\
Over                &                  &                  \\
H0=No               &                  &                  \\
test                &                  &                  \\
test                &                  &                  \\
\bottomrule
```

```
\end{longtable}
```

L'ambiente *tabu* 9

Il creatore del pacchetto `tabu`, Florent Chervet, sembra che abbia cessato di curare e aggiornare il pacchetto dal 2011. Sembra, però, che continui ad aggiornare e modificare il pacchetto sul suo sito https://github.com/tabu-fixed/tabu. Questi aggiornamenti più recenti sono però riportati nella documentazione del pacchetto del 2019.

L'autore dice che il pacchetto `tabu` mette a disposizione dell'utente gli ambienti *tabu* e *longtabu*; quest'ultimo è basato su *longtable* ma con quasi tutte le ulteriori funzionalità di *tabu*.

L'ambiente *tabu* contiene in sé tutte le funzionalità di *tabular*, *tabular**, *tabularx* e del pacchetto `array`; è compatibile con il pacchetto `siunitx` e accetta la sua colonna di tipo `S`, che serve per incolonnare i numeri fratti sulla base della cifra delle unità. Inoltre, gli ambienti sono compatibili con `booktabs`

Richiede l'uso di compilatori 'moderni', cioè che incorporano le funzionalità estese del motore di composizione ϵ-TEX. Queste funzionalità estese fanno parte di pdftex dal 2005, e sono parte integrante di xetex e luatex fin da quando esistono. Quindi non dovrebbero esserci più difficoltà di nessun genere oggigiorno, quando sono passati almeno 13 anni dall'introduzione di quelle nuove funzionalità. Chi scrive, abituato ad usare solamente versioni aggiornatissime del sistema TEX, non ha mai avuto problemi di alcun genere nell'usare i programmi che dispongono delle funzionalità estese; forse si è anche abituato ad abusarne...

Data la molteplicità di funzionalità di questo pacchetto, la lettura della sua documentazione e dei suoi molti esempi d'uso, diventa essenziale.

Gli ambienti disponibili sono i seguenti.

`{tabu}` È l'ambiente di base che funziona sia in ambiente testuale, comportandosi come *tabular*, sia in ambiente matematico, comportandosi

come *array*. Quando c'è il tipo di colonna `X` ci sono altre possibilità.

`{tabu} TO` ⟨*larghezza*⟩ si comporta come *tabular** in modo testo, ma l'estensibilità delle celle è automatica.

`{tabu} SPREAD` ⟨*dimensione*⟩ Costituisce una nuova funzionalità che permette di allargare una data tabella della quantità ⟨*dimensione*⟩; vale a dire che se con `spread 0pt` la tabella ha una data larghezza naturale, con `spread 10mm`, la tabella viene ad avere una larghezza complessiva di 10mm superiore.

`{longtabu}` Serve per comporre lunghe tabelle in modo simile a quanto si ottiene con *longtable* ma disponendo delle funzionalità estese di `tabu`.

Nella lista dei ⟨*descrittori delle colonne*⟩, sempre necessaria con qualunque tipo di ambiente per produrre tabelle, ci sono alcuni descrittori che hanno funzionalità estese rispetto a tutti i descrittori degli ambienti del kernel di LaTeX e di quelli descritti nei capitoli precedenti, nonché di quelli aggiunti dai vari pacchetti già citati.

`|[`⟨*spessore*⟩`,`⟨*colore*⟩`]` permette di specificare sia lo ⟨*spessore*⟩ sia il colore dei filetti verticali. In altre parole il descrittore `|` accetta un argomento facoltativo racchiuso fra parentesi quadre.

`X[`⟨*coef*⟩`,`⟨*allin*⟩`,`⟨*tipo*⟩`]` Il descrittore `X` permette di determinare la larghezza di questo tipo di colonne in modo da raggiungere la dimensione specificata per la tabella come con l'ambiente *tabularx*, ma accetta diversi argomenti facoltativi: se ci sono più colonne di tipo `X`, il coefficiente ⟨*coef*⟩ permette di distribuire l'allargamento in modo diverso fra le varie colonne; l'allineamento ⟨*allin*⟩ del contenuto della colonna `X` può essere a sinistra (`l`) o a destra (`r`) o può essere centrato (`c`) oppure, infine, giustificato (`j`); il ⟨*tipo*⟩ di allineamento col contenuto delle celle adiacenti può essere centrato sull'asse matematico (`m`), sulla prima riga (`p`, valore preimpostato) o sull'ultima riga (`b`); vi si riconoscono gli stessi codici dei descrittori `m, p, b` definiti dal pacchetto `array` e nel kernel.

`X[`⟨*coef*⟩`,`⟨*allin*⟩`,`⟨*tipo*⟩`,`⟨*$*⟩`]` Si comporta come nel caso precedente, ma ⟨*$*⟩ indica che la colonna contiene formule, non testo: il valore di questo parametro può essere un solo dollaro per inserire formule in `\textstyle`, oppure un doppio dollaro per inserire formule in `\displaystyle`.

`X[`⟨*opzioni per X*⟩`]{`⟨`S[`⟨*opzioni per S*⟩`]`⟩`}` Permette di creare una colonna, simile al tipo `X`, ma con le funzionalità del tipo `S`, sia generico sia con le sue opzioni.

Ci sono altri comandi possibili, da inserire dentro la lista dei ⟨*descrittori delle colonne*⟩, che qui si tralasciano, ma quelli elencati, già da soli, mostrano quanto siano vaste le configurazioni che si possono avere con un solo ambiente. Tra le altre cose, è possibile salvare l'intera lista dei descrittori con un nome, per riutilizzarla in altre tabelle con la stessa struttura.

Le righe delle tabelle possono venire colorate, anche con sequenze di due o più colori che si susseguono nell'ordine.

Esempio 9.1 ***Colonne*** `X` ***diverse***

Creiamo una tabella larga quanto la giustezza di questo testo, con celle di tipo `X` a cui assegniamo coefficienti diversi e allineamenti diversi. Mettiamo i filetti verticali per marcare visivamente la larghezza delle colonne; inseriamo anche dei filetti orizzontali in ogni cella, in modo che marchino la larghezza della cella, senza tenere conto degli spazi di separazione dai filetti verticali.

```
\noindent\begin{minipage}{\textwidth}
\vspace*{\baselineskip}
\centering
\captionof{table}{Tabella con colonne
  di tipo \descrittorestyle{X} diversamente proporzionate}
\bigskip

\begin{tabu} to \linewidth{%
  | X[1,l] | X[2,c] | X[3,c] | X[2,c] | X[1,r] |}
Testo in bandiera
  & Testo centrato
    & Testo centrato
      & Testo centrato
        & Testo in bandiera \\
\hrule & \hrule & \hrule & \hrule & \hrule \\
\end{tabu}
\vspace*{\baselineskip}
\end{minipage}
```

Tabella 9.1 Tabella con colonne di tipo X diversamente proporzionate

Testo in bandiera	Testo centrato	Testo centrato	Testo centrato	Testo in bandiera

Verifichiamo le larghezze; la giustezza della gabbia vale $352,0\mathrm{pt} \approx 128\mathrm{mm}$. Da questa larghezza totale, avendo 5 celle, dobbiamo togliere i separatori di $6\mathrm{pt}$, uno da ciascun lato del contenuto di ciascuna cella (quindi due separatori per ogni cella): in totale $60\mathrm{pt}$; inoltre dobbiamo togliere lo spessore dei 6 filetti verticali, ciascuno di $0,4\mathrm{pt}$; ci restano solamente $289,6\mathrm{pt} \approx 101,8\mathrm{mm}$. Ora dobbiamo suddividere questa lunghezza fra le cinque celle con coefficienti di larghezza diversi:

$$1x + 2x + 3x + 2x + 1x = 9x = 289,6\mathrm{pt}$$

Dividendo la lunghezza disponibile per 9, troviamo che la prima e la quinta cella sono larghe $x = 32,178\mathrm{pt} \approx 11,31\mathrm{mm}$; la seconda e la quarta sono larghe il doppio della prima, cioè $2x = 64,356\mathrm{pt} \approx 22,62\mathrm{mm}$; la terza è larga il triplo della prima, cioè $3x = 96,533\mathrm{pt} \approx 33,93\mathrm{mm}$.

Se si legge questa documentazione stampata si può verificare con un righello, ricordandosi tenere conto almeno dei $4\mathrm{mm}$ circa degli spazi fra il contenuto delle celle e i filetti; altrimenti, si possono misurare le righe orizzontali sotto il testo, che sono esattamente larghe x, $2x$ e $3x$. Se si legge a schermo, prima bisogna dimensionare l'immagine della pagina in modo che sia larga esattamente $176\mathrm{mm}$ (la base di un foglio di carta in formato B5) e poi si può appoggiare il righello sullo schermo.

Esempio 9.2 ***Una lunga tabella con*** *longtabu*

Nelle precedenti versioni di questa guida esisteva questo paragrafo che mostrava un esempio con cui si sarebbe potuto usare l'ambiente *longtabu* per generare il lungo esempio di nomenclatura svolto con gli altri ambienti descritti sopra.

Purtroppo bisogna dare addio all'ambiente *longtabu* perché, con alcune recenti modifiche (2020 – 2021) del kernel di LaTeX e dei pacchetti della

cartella `tools`, questo ambiente non è più compatibile con *longtable*, sul quale si appoggia.

Siccome il pacchetto `tabu` viene aggiornato nel suo repository originale, ma la sua documentazione non viene più modificata dal 2011, è possibile che l'autore non sia più interessato a mantenerne la compatibilità con il sistema TEX. I responsabili di CTAN, nonostante ciò, cercano di distribuire il pacchetto `tabu` come aggiornato nel suo repository originale, ma non sono responsabili del mantenimento della compatibilità. Ne hanno informato l'autore, ma a quanto pare, senza nessun esito.

L'ambiente *tabbing* 10

L'ambiente *tabbing* produce una tabulazione come si sarebbe fatto con le vecchie macchine da scrivere, cioè si impostano gli arresti di tabulazione e in ogni riga da scrivere si indica di passare al successivo arresto con opportuni codici; insomma qualcosa del genere:

```
\begin{tabbing}
⟨imposta gli arresti di tabulazione⟩\kill
⟨testo⟩ \> ⟨testo⟩ \> ... \> ⟨testo⟩\\
⟨testo⟩ \> ⟨testo⟩ \> ... \> ⟨testo⟩\\
...
⟨testo⟩ \> ⟨testo⟩ \> ... \> ⟨testo⟩\\
\end{tabbing}
```

La tabulazione accetta i fine pagina, quindi può proseguire per diverse pagine.

L'ambiente è fragile, per cui non può essere messo nell'argomento di altri ambienti o comandi che spostino i loro argomenti da una macro all'altra. In particolare un ambiente *tabbing* non può essere annidato dentro un altro ambiente *tabbing*, al contrario di *tabular* e di altri suoi simili che possono essere tranquillamente annidati.

Per ovviare a tale inconveniente, l'ambiente *tabbing* gestisce uno *stack* nel quale può spingere (*push*) una data impostazione di arresti di tabulazione, ne può impostare un'altra e, dopo alcune righe composte con questa nuova impostazione, può richiamare (*pop*) la precedente e continuare a comporre con gli arresti originali.

Naturalmente, comporre con un programma di composizione non è la stessa cosa dello scrivere a macchina, quindi l'ambiente *tabbing* offre qualcosa in più dei semplici arresti di tabulazione.

Prima di fare esempi, conviene specificare come si impostano gli arresti di tabulazione e come si passa da un arresto al successivo.

`\=` Imposta gli arresti di tabulazione o nella prima riga della tabulazione stessa o, meglio, in una riga iniziale a sé stante terminata con il comando `\kill`.

`\kill` Termina una riga di impostazione degli arresti, memorizzando le loro posizioni, ma non scrivendo nulla nel documento di quanto compare nella riga di impostazione. In sostanza è opportuno impostare gli arresti in questo modo:

⟨*spazio*$_1$⟩`\=`⟨*spazio*$_2$⟩`\=...\kill`

dove i vari spazi sono stringhe di testo (di cui *tabbing* calcola la larghezza) o comandi di spaziatura orizzontale; se si tratta di stringhe, allora è bene scegliere la stringa di ⟨*testo*⟩ più lunga per quella colonna di tabulazione; se si tratta di spazi si usi il comando `\hspace*{`⟨*spazio*⟩`}` che non viene mai eliminato dal programma di composizione; non è necessario nulla dopo l'ultimo comando di impostazione che compare nella riga "uccisa" con `\kill`.

`\>`⟨*testo*⟩ Sposta l'inizio della composizione del ⟨*testo*⟩ da tabulare al successivo arresto di tabulazione.

`\\` Come altrove, termina la riga e ne comincia un'altra dalla posizione dell'arresto iniziale memorizzato con `\+`; di default, senza mai avere usato `\+`, l'arresto iniziale (l'arresto numero 'zero') coincide con l'inizio della griglia di stampa.

`\+` Incrementa di una unità il valore corrente del contatore di tabulazione, quindi da questo momento in poi ogni riga comincia non dall'inizio della griglia, ma dall'i-esimo arresto.

`\-` Decrementa di un'unità il valore corrente del contatore degli arresti, riportando indietro l'inizio della riga composta.

`\<` Sposta indietro di una posizione di tabulazione; può essere usato solo all'inizio di una nuova riga.

`\'`⟨*testo*⟩ Sposta a destra un intero ⟨*testo*⟩ a filo del successivo arresto di stampa

`\‘`⟨*testo*⟩ Sposta a sinistra un intero ⟨*testo*⟩ a filo del precedente arresto di stampa.

`\pushtabs` Spinge in memoria l'attuale definizione degli arresti di stampa, consentendo di definirne un'altra serie.

`\poptabs` Recupera dalla memoria l'ultima definizione di arresti di stampa che erano stati precedentemente memorizzati.

`\a=` `\a'` e `\a‘` Questi tre comandi sono un residuo storico di quando il sistema TEX accettava solo caratteri ASCII e gli accenti; in particolare, il segno di 'lunga', l'accento acuto e l'accento grave venivano introdotti con i comandi `\=`, `\'` e `\‘`, che dentro l'ambiente *tabbing* hanno un altro significato.
Ora, con la possibilità di usare codifiche di vario genere, compresa la codifica *utf8*, ci pensa il gestore della codifica a distinguere la funzione e a usare il comando giusto per gli accenti all'interno della LICR (LATEX Internal Character Representation), quindi l'utente non ha più bisogno di servirsi di questa serie di comandi per usare gli accenti all'interno di un ambiente *tabbing*; tuttavia, è importante che sappia che esistono per servirsene in quei rari casi in cui fosse necessario introdurre una lettera accentata 'esotica'.[1]

Va inoltre detto che oggigiorno è difficile dover ricorrere a questo ambiente *tabbing*; chi scrive ricorda di averlo usato per l'ultima volta diversi anni fa (2015). Tuttavia, non è escluso che in qualche occasione non sia possibile o necessario usarlo.

Esempio 10.1 ***Composizione con*** *tabbing*

In un manuale didattico di diverse materie è capitato di dover mettere l'elenco dei curatori delle diverse discipline; chi scrive non si ricorda perché una delle soluzioni da eseguirsi con le tabelle della classe usata per quel libro non andassero bene, tuttavia la soluzione con l'ambiente *tabbing* risultò adatta allo scopo. Il codice usato fu il seguente:

[1] Con lualatex e xelatex non ci sono problemi, se non l'eventuale uso di driver di tastiera diversi da quello per la tastiera italiana; con pdflatex potrebbero esserci dei problemi; se per esempio si volesse indicare il nome del creatore del programma pdftex, Hàn Thế Thành (vietnamita), sarebbe necessario sovrapporre due accenti, perché i caratteri vietnamiti non sono presenti nella solita codifica *T1*. Certo, si potrebbe invocare il modulo di `babel` per la lingua vietnamita, che usa una codifica diversa, ma per un nome solo non ne vale la pena. Ecco quindi che per comporre la lettera ế, bisogna usare accenti sovrapposti o usare la macro per l'accento grave su una lettera già accentata.

```
\begin{tabulazione}
\caption{Incolonnamento con \ambstyle{tabbing}}
\label{tab:con tabbing}
\begin{tabbing}
Scienze fisiche e chimiche: \=\kill
Curatori:\\[5mm]
Logica:                \> \textsc{Pierluigi Neri},
                          \textsc{Massimo Verdi},\\
                       \> \textsc{Gisella Bruni},
                          \textsc{Cristiana Gialli}\\[5mm]
Comprensione verbale: \> \textsc{Diana Violetti},
                          \textsc{Chiara Arancini}\\[5mm]
Matematica:            \> \textsc{Laura Nerini},
                          \textsc{Giovanna Bicolore}\\
                       \> \textsc{Marcello Grigetti},
                          \textsc{Marco Ambrati}\\[5mm]
Scienze fisiche e chimiche:
                       \> \textsc{Giovanni Bianchi},
                          \textsc{Antonio Rossi}
\end{tabbing}
\end{tabulazione}
```

Il risultato della compilazione è riportato nella tabulazione 10.1. L'ambiente *tabulazione* differisce dall'ambiente *table* solo per l'etichetta della didascalia.

Esempio 10.2 ***Altra tabulazione***

In questo secondo esempio, gli arresti di tabulazione sono specificati mediante spazi, invece che con stringhe. Il codice è il seguente, e il risultato appare nella tabulazione 10.2.

```
\begin{tabulazione}
\caption{Incolonnamento con tre colonne}
\label{tab:tabulazione con città}
\begin{tabbing}
\hspace{50mm} \=\hspace{45mm}\=\kill
Logica: \>\textsc{Pierluigi Neri}    \> Firenze\\
        \>\textsc{Massimo Verdi}     \> Pisa\\
```

Tabulazione 10.1 Incolonnamento con *tabbing*

Curatori:

Logica:	Pierluigi Neri, Massimo Verdi, Gisella Bruni, Cristiana Gialli
Comprensione verbale:	Diana Violetti, Chiara Arancini
Matematica:	Laura Nerini, Giovanna Bicolore Marcello Grigetti, Marco Ambrati
Scienze fisiche e chimiche:	Giovanni Bianchi, Antonio Rossi

```
        \>\textsc{Luisella Bruni}    \> Siena\\
        \>\textsc{Cristiana Gialli}  \> Ancona\\[5mm]
Comprensione verbale:
        \>\textsc{Diana Violetti}    \> Perugia\\
        \>\textsc{Chiara Arancini}   \> Genova \\[5mm]
Matematica:
        \>\textsc{Laura Nerini}      \> Torino\\
        \>\textsc{Giovanna Bicolore} \> Napoli\\
        \>\textsc{Marcello Grigetti} \> Milano\\
        \>\textsc{Marco Ambrati}     \> Bologna\\[5mm]
Scienze fisiche e chimiche:
        \>\textsc{Giovanni Bianchi}  \> Padova\\
        \>\textsc{Antonio Rossi}     \> Trieste
\end{tabbing}
\end{tabulazione}
```

È chiaro che esempi così semplici non mettono in luce tutte le potenzialità dell'ambiente *tabbing*. Ma di solito, per tabulazioni più complesse si ottengono migliori risultati con gli ambienti per costruire tabelle.

Tabulazione 10.2 Incolonnamento con tre colonne

Logica:	Pierluigi Neri	Firenze
	Massimo Verdi	Pisa
	Luisella Bruni	Siena
	Cristiana Gialli	Ancona
Comprensione verbale:	Diana Violetti	Perugia
	Chiara Arancini	Genova
Matematica:	Laura Nerini	Torino
	Giovanna Bicolore	Napoli
	Marcello Grigetti	Milano
	Marco Ambrati	Bologna
Scienze fisiche e chimiche:	Giovanni Bianchi	Padova
	Antonio Rossi	Trieste

Tabelle, versione v.1.1.02 del 2021-09-22

Il pacchetto `tabularray` 11

Il 1º settembre 2021 CTAN si è arricchito di un nuovo potentissimo pacchetto per costruire tabelle e matrici, `tabularray`' l'autore si è ispirato ai pacchetti citati nei paragrafi precedenti, producendo le stesse funzionalità con un unico pacchetto e aggiungendo molte altre funzionalità.

Preciso subito che la sua documentazione, pur sintetica, contiene 156 pagine; virtualmente in ogni pagina c'è almeno un esempio d'uso; questo dà un'idea di quante cose questo pacchetto sia capace di fare.

Il suo punto forte è che è basato sul linguaggio LaTeX3; quindi i comandi presentano diverse funzionalità a seconda della configurazione che viene specificata per ciascuno di essi.

L'utente può imporre al suo unico ambiente *tblr* (acronimo di "tabularray", ma anche sigla formata con l'unione di alcuni descrittori di colonna) delle opzioni sia globali sia locali; il pacchetto può colorare sia le righe, sia le colonne, sia i filetti; per questi può specificare lo spessore sia globalmente sia localmente. I contenuti delle celle possono essere allineati verticalmente con diversi codici che funzionano correttamente sia con gruppi di caselle della stessa riga, sia con quelli della stessa colonna. Le cose funzionano correttamente anche quando la tabella o la matrice contiene simultaneamente combinazioni strane di raggruppamenti di righe e colonne. Lo stesso ambiente con le stesse opzioni funziona sia in ambiente testo, sia in display, sia in ambiente matematico; in sostanza quell'unico ambiente ha le funzionalità di *tabular*, *tabular**, *array*, *tabularx*, *widetabular*, *tabu*; esso svolge anche i compiti di *longtable* e *supertabular*.

Esempio 11.1 ***Il titolo della documentazione di*** `tabularray`

Cominciamo subito con un esempio, preso dalla composizione del titolo della documentazione di *tabularray* con lievi modifiche.

La tabella 11.1 è costruita con il codice seguente:

Tabella 11.1 Il titolo della documentazione di `tabularray`

Tabularray	**Typeset Tabulars and Arrays with LaTeX3**
Author	Jianrui Lyu (tolvjr@163.com)
Version	2021N (2021-09-01)
Code	https://github.com/lvjr/tabularray
Code	https://bitbucket.org/lvjr/tabularray
Forum	https://github.com/lvjr/tabularray/discussions
Forum	https://tex.stackexchange.com/questions/tagged/tabularray
Issue	https://github.com/lvjr/tabularray/issues

```
\begin{tblr}{
  colspec = {rX}, colsep = 8mm, hlines = {2pt, white},
  row{odd} = {azure8}, row{even} = {gray8},
  row{1} = {6em,azure2,fg=white,font=\LARGE\bfseries\sffamily},
  row{2-Z} = {3em,font=\Large},
}
Tabularray & Typeset Tabulars and Arrays with \LaTeX3 \\
Author     & Jianrui Lyu (tolvjr@163.com) \\
Version    & 2021N (2021-09-01) \\
Code       & \url{https://github.com/lvjr/tabularray} \\
Code       & \url{https://bitbucket.org/lvjr/tabularray} \\
Forum      & \url{https://github.com/lvjr/tabularray/discussions} \\
Forum      & \url{https://tex.stackexchange.com/questions/%
             tagged/tabularray} \\
Issue      & \url{https://github.com/lvjr/tabularray/issues} \\
\end{tblr}
```

Si nota subito che la riga modello della tabella è sostituita da un "preambolo" piuttosto dettagliato.

1. Vi si vedono le specifiche delle colonne impostate con l'opzione *colspec*; le due colonne sono un allineamento a destra e una colonna di tipo `X`, esattamente come nell'ambiente *tabularx*.

2. Seguono le specifiche dei separatori di colonna, e dei filetti orizzontali, di cui viene specificato sia lo spessore sia il colore; siccome le righe sono colorate, il fatto che i filetti siano bianchi non è un modo per nasconderli.

3. Le righe dispari sono colorate di azzurro e quelle pari di grigio; i colori numerati derivano da un pacchetto che l'ambiente si carica di default, ma l'utente può caricare il pacchetto `xcolor` e può usare i colori che vuole.

4. Per la riga 1, la riga dei titoli delle colonne, oltre ad una spaziatura e ad una diversa tonalità di colore azzurro, viene specificato il colore bianco del testo e le caratteristiche del font da usare. Viene specificata una spaziatura e un font diverso

5. Per le righe dalla 2 alla fine Z, oltre ad una spaziatura, viene specificato un font di corpo minore rispetto alla riga dei titoli

Tabella 11.2 una tabella con celle che si sviluppano contemporaneamente su più righe e più colonne

r=3 c=2		r=2 c=3
	MIDDLE	
r=2 c=3		r=3 c=2

6. Nelle specifiche di tutte le righe la spaziatura indicata indica l'altezza della riga. Senza diverse specificazioni, il contenuto della cella viene centrato verticalmente dentro ciascuna cella.

Il "preambolo" sembra prolisso; in realtà è molto più chiaro di quello che si sarebbe dovuto usare con gli ambienti disponibili prima dell'avvento di questo pacchetto.

Esempio 11.2 ***Una tabella con righe e colonne che si sviluppano contemporaneamente su più righe e più colonne***

Prendo un altro esempio dalla documentazione di `tabularray`.

Il preambolo della tabella 11.2è semplicissimo e non differisce da quello che si sarebbe usato con l'ambiente *tabular*; ma il corpo della tabella contiene delle specifiche particolari per i filetti; compare anche il comando `\SetCell` usato per descrivere ciascuna cella; si noti che la tabella contiene tre righe e cinque colonne; i codici che contengono le sigle 'r=' e 'c=' ricordano quante righe e quante colonne sono coinvolte da ciascuna cella.Il codice usato è il seguente:

```
\begin{tblr}{|ll|c|rr|}
\hline
 \SetCell[r=3,c=2]{h} r=3 c=2 & 1-2 &
    \SetCell[r=2,c=3]{r} r=2 c=3 & 1-4 & 1-5 \\
 2-1 & 2-2 & 2-3 & 2-4 & 2-5 \\
\hline
 3-1 & 3-2 & MIDDLE & \SetCell[r=3,c=2]{f} r=3 c=2 & 3-5 \\
\hline
 \SetCell[r=2,c=3]{l} r=2 c=3 & 4-2 & 4-3 & 4-4 & 4-5 \\
 5-1 & 5-2 & 5-3 & 5-4 & 5-5 \\
\hline
\end{tblr}
```

1. L'argomento facoltativo di `\SetCell` è il testo da inserire nella cella che ne occupa due o tre fra quelle specificate nel preambolo; L'argomento obbligatorio indica l'allineamento verticale o orizzontale di tale contenuto: `h` indica che l'allineamento verticale è fatto rispetto alla prima (o unica) riga del contenuto; `r` indica che il contento à allineato a destra; `l` indica che il contenuto è allineato a sinistra; `c` indica che il contenuto è centrato orizzontalmente nella cella; ogni argomento obbligatorio potrebbe contenere due descrittori, uno verticale e uno orizzontale.

2. la grossa novità è che i filetti orizzontali non compaiono nelle celle condivise dalle "multicelle" specificate; quindi in questi casi non è necessario specificare in quali colonne i filetti vanno disegnati.

3. Analogamente la riga che contiene la parola 'MIDDLE', a alla sua sinistra due campi *non vuoti*, che però non appaiono nella tabella composta perché occuperebbero la multicella verticale alla sua sinistra.

4. Invece, cioè è scritto a destra di 'MIDDLE', è la specifica di una multicella il cui contenuto è allineato al fondo col descrittore `f`.

5. Cose analoghe avvengono nella terza riga a destra della prima multicella.

Ma allora a che cosa serve scrivere qualcosa nelle celle che non compaiono perché sovrapposte con multicelle? Né più né meno, come fossero dei commenti, che permettono, per esempio, di localizzare le celle nel codice sorgente, anche se nella colonna composta non appaiono.

Esempio 11.3 ***Evidenziare una formula con colori diversi per la cornice, lo sfondo, e il testo***

La versatilità di `tabularray` si può applicare anche ad usi insoliti. Per esempio sul forum del GUIT è apparsa la richiesta di comporre una equazione molto in evidenza, cioè con una cornice di adeguato spessore e colorata; con lo sfondo colorato e con la formula scritta con un terzo colore. Il pacchetto `tabularray` permette di specificare i colori egli spessori per ciascuna delle caratteristiche richieste; l'equazione 11.1 appare appunto una formula evidenziata come specificato: cornice fatta di filetti spessi 2pt e colorati di rosso; formula blu su uno sfondo giallo.

$$\boxed{F(s) = \int_{0-}^{+\infty} f(t)\,\mathrm{e}^{-st}\,\mathrm{d}t} \tag{11.1}$$

Il codice per comporre la formula 11.1 è il seguente:

```
\begin{equation}
\begin{tblr}{
  colspec = {| c |},
  hlines = { 2pt, red }, vlines = {2pt, red},
  row{1-Z} = { 4em, yellow, fg=blue }
}\displaystyle
F(s) = \int_{0-}^{+\infty} f(t) \eu^{-st}\diff t
\end{tblr}
\label{equ:formula colorata}
\end{equation}
```

Sono opportuni alcuni commenti.

1. Come si è già specificato, l'ambiente *tblr* funziona indifferentemente in modo testo (come *tabular* e i suoi simili) e in modo matematico (come *array*, ma senza gli errori di quest'ultimo).

2. Va segnalato, però, che in modo matematico i descrittori delle colonne possono essere solo quelli che anche *array* accetta, vale a dire, solo `c`, `l` e `r`, e non funziona correttamente con i descrittori che presuppongono per le loro celle solo del testo. Quindi non bisogna usare i descrittori `p`, `m`, `b`, `X` e simili.

3. Come per tutte le matrici, le celle dell'ambiente *tblr* sono composte in `\textstyle`, quindi se l'utente, come per la formula 11.1, vuole comporre il "testo" matematico in `\displaystyle`, lo deve esplicitare.

4. Si segnala che i comandi `\eu` e `\diff` compongono i rispettivi simboli 'e' e 'd' in tondo, secondo le norme UNI; in fondo la formula 11.1 definisce la trasformazione di Laplace, base di ogni lavori tecnico-scientifico nei domìni dell'elettrotecnica, dell'elettronica e delle telecomunicazioni. Poiché questa guida è composta con pdfLaTeX, si sono usati i simboli del pacchetto `pm-isomath`.

Questi esempi probabilmente sono sufficienti a mostrare la potenza e la versatilità del pacchetto `tabularray`; esso è in grado di fare anche altre cose, molto difficili da ottenere con gli altri ambienti descritti nei capitoli precedenti. Si rinvia il lettore alla documentazione del pacchetto, dando da terminale il comando `texdoc tabularray`.

Considerazioni finali

In questa guida sono descritti i modi per comporre belle tabelle secondo la *best practice* dell'arte tipografica. Sono mostrate anche tabelle da *non* comporre per niente, allo scopo di mostrare alcune particolarità che di solito non sono così evidenti; in particolare, l'uso dei filetti verticali è mostrato pragmaticamente per vedere che cosa succede davvero se li si usa. Quasi sempre non sono necessari, spesso confondono il lettore, in ogni caso non fanno parte della *best practice*.

Il pacchetti citati sono quelli che chi scrive ha usato effettivamente; li ha anche commentati e ha spiegato perché ha orientato le sue scelte personali verso alcuni e non verso altri pacchetti. Egli ribadisce qui che si tratta essenzialmente di gusti personali oltre che di una maggiore pratica acquisita con i pacchetti che preferisce.

Non dimentichiamo però che LaTeX è un linguaggio di mark up ed è anche un linguaggio di programmazione; lascia libero ogni utente di esercitare la fantasia per inventarsi soluzioni personali per risolvere i propri problemi di composizione. L'unica cosa che l'utente deve osservare non sono i vincoli alla fantasia, ma l'osservanza delle norme poco o per nulla scritte in merito alla tipografia; i bravi tipografi sono artisti e sentono come e quando possono violare creativamente le consuetudini della loro arte. La maggior parte di noi, che usiamo LaTeX con passione, è formata da dilettanti; quindi il dilettante deve documentarsi sull'arte tipografica oltre che sviluppare la capacità di programmare.

E, appunto, per programmare bene con LaTeX, è necessario documentarsi. Tutti i pacchetti citati in questa guida, come si è già osservato, sono documentati e forniti assieme all'installazione del sistema TeX (completo e aggiornato), sia esso installato con la distribuzione di TeX Live oppure con quella di MikTeX. Per ogni pacchetto basta aprire un terminale o un prompt dei comandi, comunque si chiami nella macchina dell'utente

e del sistema operativo che la controlla: vi si scrive dentro il comando `texdoc` seguito dal nome del pacchetto, si preme il tasto `Invio`, e si apre subito una finestra con la documentazione; la si deve studiare attentamente, cercando di capirne la logica, e facendone una pratica sufficiente.

Indice analitico

Simboli

A

B

C

D