Numero 29
Aprile 2020

ArsTEXnica

Rivista italiana di TEX e LATEX

GUIT

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

TEXnica Ars

GUIT – Gruppo Utilizzatori Italiani di TEX

ArsTEXnica è la pubblicazione ufficiale del GUIT

ArsTEXnica è la prima rivista italiana dedicata a TEX, a LATEX e alla tipografia digitale. Lo scopo che la rivista si prefigge è quello di diventare uno dei principali canali italiani di diffusione di informazioni e conoscenze sul programma ideato da Donald Knuth.

Le uscite hanno cadenza semestrale e sono pubblicate nei mesi di Aprile e Ottobre. In particolare, la seconda uscita dell'anno contiene gli Atti del Convegno Annuale (GUIT *meeting*).

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

arstexnica@guitex.org.

Pubblicare su ArsTEXnica

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

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

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

Nota sul Copyright

Associarsi a GUIT

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

Indirizzi

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

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

ISSN 1828-2350 (Stampa)
ISSN 1828-2369 (Online)

15 Aprile 2020

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 29, Aprile 2020

Gruppo Utilizzatori Italiani di TEX

Editoriale

Francesco Biccari

Questo numero di ArsTEXnica segna una svolta. Il 7 aprile 2020 è stata riconosciuta la *scientificità* di ArsTEXnica da parte dell'Agenzia Nazionale di Valutazione del sistema Universitario e della Ricerca (ANVUR).

Ma partiamo dall'inizio. Da qualche anno, a seguito della legge n. 240/2010 (parte della Riforma Gelmini), uno dei requisiti per essere assunti come professori in un'università italiana è il possesso dell'Abilitazione Scientifica Nazionale (ASN), una idoneità ottenuta presentando domanda al Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR). Per ottenerla si devono rispettare diversi criteri, il più importante dei quali è l'aver pubblicato un certo numero di articoli nel settore per cui si fa richiesta. Gli articoli che possono essere dichiarati per i settori prettamente scientifici come fisica, matematica, ecc... (*settori bibliometrici*) sono quelli che vengono indicizzati da due famosi database internazionali, Scopus e Web of Science, su cui il MIUR non ha controllo. Ci sono poi le scienze umane e sociali (*settori non bibliometrici*) che per loro natura sono più legate alla realtà italiana e quindi non trovano posto, o lo trovano solo in parte, in questi database internazionali. Per questo motivo il MIUR ha incaricato l'ANVUR di mantenere una lista di riviste italiane che pubblicano in questi settori e i cui articoli sono validi ai fini della richiesta dell'ASN. Il 29 maggio 2019 il GuIT ha presentato l'istanza di classificazione di ArsTEXnica come rivista scientifica nei *settori non bibliometrici* di area 10 (scienze dell'antichità, filologico-letterarie e storico-artistiche) e di area 11 (scienze storiche, filosofiche, pedagogiche e psicologiche). La *scientificità* in entrambe le aree è stata riconosciuta il 7 aprile 2020 ed è valida per i numeri pubblicati dal 2016 in poi.[1] Questo risultato, oltre al prestigio per il GuIT, dovrebbe portare un numero maggiore di contributi ad ArsTEXnica.

Un'altra novità è stata il rinnovo di tutte le cariche del GuIT. Il 1º marzo 2020 si è insediato il nuovo Consiglio Direttivo, guidato dal nuovo Presidente, il prof. Guido Milanese. Le idee e le iniziative sul tavolo sono molte. Si percepisce una grande energia e voglia di far crescere il GuIT. Un ringraziamento va al vecchio Consiglio e un augurio di buon lavoro al nuovo.

Insieme al Consiglio è stato rinnovato di conseguenza anche il Comitato Editoriale di ArsTEXnica. Rimangono le colonne portanti, i coordinatori di redazione Massimiliano Dominici e Gianluca Pignalberi, a cui si aggiunge la consigliera Silvia Maschio. Il prof. Claudio Beccari, a cui vanno gli auguri di tutto il GuIT per i suoi 80 anni compiuti lo scorso 28 aprile, ha purtroppo deciso di lasciare la direzione della rivista per dare spazio alle nuove leve. E così, dopo la fiducia accordatami dal Consiglio, sono diventato il nuovo direttore. Non potrò mai essere all'altezza del nostro Claudio, ma mi impegnerò al massimo affinché ArsTEXnica sia sempre più nota e attrattiva, sia per autori italiani che stranieri, e sia rinnovata nella gestione del processo di *peer-review* degli articoli, con il coinvolgimento di revisori esterni, già dal numero attuale.

Dopo questa serie di notizie istituzionali, vediamo gli articoli che troverete in questo numero.

Apriamo con un articolo di Claudio Beccari, da sempre un grande appassionato dell'ambiente pictu-re di LaTeX, che ci parla di come ha usato le nuove funzionalità offerte da LaTeX3 per estendere alcuni comandi del suo pacchetto curve2e. In particolare, vengono mostrati degli esempi per disegnare, grazie a queste nuove funzionalità, molte costruzioni della geometria euclidea, che sono poi confluiti nel nuovo pacchetto euclideangeometry.

Gianluca Pignalberi ci confeziona, è il caso di dirlo, un originale articolo per realizzare dei *pattern*, tipicamente adatti per carte regalo, con una moltitudine di approcci diversi.

Tommaso Gordini, traducendo parte della documentazione del pacchetto siunitx di Joseph Wright, ci mostra le strategie utili a comporre tabelle ricche di numeri e unità di misura.

Chiudiamo con un breve articolo di Claudio Beccari e Gianluca Pignalberi in cui si mettono a confronto due pacchetti per il disegno vettoriale, TikZ e curve2e, prendendo spunto dalla necessità di comporre il logo della casa editrice CLUT.

Vi auguro una buona lettura, dandovi appuntamento al numero autunnale che spero sarà frutto di un GuIT*meeting* dove saremo presenti fisicamente, a significare che questo difficile momento, che stiamo vivendo a causa del virus SARS-CoV-2, sarà solo un brutto ricordo.

▷ Francesco Biccari
Dipartimento di Fisica e Astronomia
Università degli Studi di Firenze
Firenze, Italia
biccari@gmail.com

1. Pagina del sito web dell'ANVUR sulla classificazione delle riviste: https://www.anvur.it/attivita/classificazione-delle-riviste/

Costruzioni di geometria euclidea mediante l'ambiente **picture** esteso

Claudio Beccari

Sommario

Il pacchetto curve2e è stato arricchito con gli aggiornamenti di LaTeX 2_ε disponibili dal 2019. Con queste ultime estensioni è possibile creare le macro necessarie per disegnare i problemi di geometria euclidea all'interno dell'ambiente picture esteso.

Abstract

The curve2e package has been enriched with the upgrades of LaTeX that took place in 2019. With such extensions it is possible to define the necessary macros to draw many problems of Euclidean geometry with the extended picture environment.

1 Introduzione

È chiaro che un programma di tipocomposizione deve poter comporre anche disegni al tratto per illustrare quanto viene descritto, specialmente, ma non solo, nei testi di tipo didattico concernenti diversi aspetti della matematica.

Fin dall'inizio, il formato LaTeX prevedeva un ambiente di base per disegnare semplici diagrammi al tratto. A quei tempi, i font vettoriali erano agli inizi e LaTeX non era attrezzato per gestirli; inoltre, i file composti erano in formato DVI (*DeVice Independent*) e quindi era impensabile richiedere a LaTeX di produrre file vettoriali scalabili con continuità, senza perdere qualità né nello scritto né nelle parti grafiche.

Nel 2003 è apparsa la prima "modernizzazione" suggerita dallo stesso Leslie Lamport nella seconda edizione del suo manuale (Lamport, 1994); in quell'anno due esperti di programmazione LaTeX hanno materialmente creato il codice col pacchetto di estensione pict2e, ora ulteriormente aggiornato (Gässlein *et al.*, 2016). Nel 2006 ho reso disponibile il pacchetto curve2e, il cui ultimo aggiornamento risale al 2020 (Beccari, 2020a); si veda anche il nuovo manuale (Beccari, 2020b).

Oltre all'ambiente picture e alle sue estensioni, nell'arco degli anni sono apparsi diversi pacchetti per il disegno programmato e/o programmi esterni capaci di esportare i loro disegni vettoriali in modo da essere compatibili con la qualità tipografica di LaTeX. Fra i secondi vorrei citare il tradizionale programma `gnuplot` (Miklavec, 2013) con interfaccia letterale, e il più moderno `asymptote` Hammerlind *et al.* (2018) con la sua interfaccia grafica.

Fra i primi, direi che non ci sia nulla di migliore della collezione che passa sotto il nome di Portable Graphic Format (PGF) (Tantau, 2019), di cui Ti*k*Z e pgfplots sono gli esponenti maggiori, diventati ormai quasi lo standard nella comunità degli utenti del sistema TeX sparsi per il mondo. Figlio del PGF, esiste anche il pacchetto tkz-euclide (Matthes, 2020).

Per gli amanti di PostScript esiste anche la collezione di pacchetti accessibili principalmente con pstricks e la sua collezione di librerie per categorie di disegni particolari (Van Zandt, 2003). Si veda anche l'articolo di De Marco (2009) su `asymptote` e De Marco (2019), che fa una bella panoramica dei vari strumenti a disposizione per il disegno di alta qualità adatto ai documenti composti con i programmi di composizione del sistema TeX.

Ma c'è anche METAPOST (Hobby, 2018), derivato da METAFONT (Knuth, 1986), quest'ultimo costruito dal padre di TeX, D.E. Knuth, per disegnare i font che lui stesso ha usato fin dagli albori di TeX nel 1978.

Perché allora dedicarsi ad ulteriori estensioni del vecchio ambiente picture?

I motivi possono essere diversi a seconda dei punti di vista; ne cito solo alcuni.

1. Il manuale di Ti*k*Z è lunghissimo; certamente le funzionalità di Ti*k*Z sono quasi illimitate, e quindi c'è molto da descrivere; ma prima di essere produttivi, bisogna studiarne a lungo il mastodontico manuale.
2. METAPOST deriva da METAFONT; il secondo disegna caratteri in un suo formato a matrici di punti; il primo esegue disegni vettoriali, volendo anche caratteri, ma, a parte la presa in carico della stessa sintassi e logica di METAFONT, non direi che sia nato solo per disegnare caratteri. Anzi, credo che sia usato più spesso per eseguire altri disegni.
3. Il manuale di pict2e, la prima estensione dell'ambiente picture, ammonta ad una ventina di pagine; la sua ulteriore estensione curve2e ha un manuale d'uso di poco più lungo. Entrambi sono abbastanza essenziali, ma usano le funzionalità del sistema TeX per definire altre macroistruzioni, per cui le funzionalità dei pacchetti possono essere ulteriormente estese dagli utenti, in modo da poter disporre di comandi personalizzati.

4. Il LaTeX 3 Team sta procedendo velocemente con la definizione del nuovo linguaggio LaTeX 3 (sigla L3). Molti moduli sono già in versione beta; quindi l'interfaccia verso l'utente è sostanzialmente stabile, anche se i dettagli delle definizioni interne possono ancora cambiare e/o le funzionalità possono aumentare.
   Fra questi moduli nuovi ce ne sono due, xparse e xfp, con caratteristiche particolarmente avanzate; il primo serve per estendere ulteriormente le definizioni di comandi e ambienti nuovi in modo da creare comandi con interfacce utente estremamente versatili. Il secondo mette a disposizione la libreria di calcoli decimali a virgola mobile (*floating point*, da cui deriva la parte 'fp' nel nome del pacchetto).
   La documentazione di xparse è completa; quella di xfp è molto (forse troppo) succinta, ma, se si legge la parte XXIII di `interface3.pdf`, si riesce a capire quali siano le funzionalità del pacchetto.
5. Il pacchetto xpicture (Fuster, 2012) è eccellente ed estende moltissimo l'ambiente picture; esso stesso usa i pacchetti pict2e e curve2e ma aggiunge molte funzionalità. In un certo senso potrebbe rimpiazzare tutto quello che viene esposto in questo articolo; ma, secondo me, la sua sintassi è un po' complessa anche se molto efficace. D'altra parte qui voglio mostrare come usare la grafica dell'ambiente picture, estesa con le funzionalità di curve2e, per costruire le procedure geometriche che consentono di risolvere graficamente alcuni problemi di geometria euclidea, con metodi non dissimili da quelli che una volta si sviluppavano con riga, squadra e compasso.

Stimolato da un articolo di Estevão Candia (Candia, 2019) su *TUGboat*, ho provato a vedere se quello che lui descrive nell'articolo, sinossi della sua tesi magistrale (Candia, 2018), sviluppato con METAPOST, si potesse fare con l'ambiente picture, quello esteso o estensibile come descritto sopra. Mi ci sono cimentato e ho ottenuto risultati simili, se non migliori, proprio con l'ambiente picture esteso. Inizialmente ho avuto qualche difficoltà perché nessuna delle estensioni di quell'ambiente consente di risolvere equazioni o sistemi di equazioni, sia pure solo lineari. METAPOST è nato per farlo, oltre che per disegnare; quindi, ovviamente, con METAPOST è marginalmente più facile ottenere i risultati desiderati, ma prima di essere capaci di usare correttamente METAPOST bisogna impararne il suo linguaggio che, però, non è semplice. Nel dir questo mi metto nei panni del normale utente di LaTeX che potrebbe trovare delle difficoltà.

Nel contempo ho imparato molte cose; per me quindi è stato utile; ma penso che sia utile anche agli insegnanti di scuola secondaria superiore che scrivono con LaTeX gli appunti o le dispense per i loro allievi. Forse possono essere utili anche per i docenti delle matematiche propedeutiche dei corsi universitari di laurea triennale.

Per questo motivo ho messo a disposizione il nuovo pacchetto euclideangeometry (Beccari, 2020c), che dispone di un manuale utente dettagliato (Beccari, 2020d) dove, tra l'altro, sono estesamente ripresi e commentati alcuni degli esempi mostrati in questo articolo. Il pacchetto dovrebbe già fare parte di ogni installazione completa e aggiornata di TeX Live e MiKTeX. L'utente interessato non ha altro che da caricare questo nuovo pacchetto, il quale provvede da solo a caricare curve2e, di cui è un'estensione; curve2e a sua volta carica pict2e cosicché la catena di estensioni dell'ambiente picture è completa.

Attenzione! Questo nuovo pacchetto euclideangeometry richiede l'uso dell'ultima versione di curve2e, con la data dell'ultima revisione non anteriore al 2020-01-18. In mancanza di questo requisito euclideangeometry emette un vistoso messaggio d'errore e termina il suo stesso caricamento oltre che il processo di compilazione. In sostanza questo pacchetto funziona solo con un'installazione completa e aggiornata del sistema TeX.

Questo articolo è diviso in due parti. La prima, di carattere generale, descrive i comandi basati sul pacchetto xfp che sono già incorporati nell'ultima versione di curve2e; sono già documentati nella descrizione d'uso (Beccari, 2020b) e in quella del codice di curve2e (Beccari, 2020a), ma merita descriverli succintamente anche qui come esempi di nuovi comandi per gestire i numeri in virgola mobile e per gestire i test da eseguire su tali numeri. Nella seconda parte descriverò come realizzare certi disegni geometrici che riguardano principalmente triangoli, poligoni regolari, circonferenze ed ellissi, in particolare quelle che si inscrivono nei triangoli.

2 Upgrade di curve2e

2.1 Retrocompatibilità

Da quando è divenuto disponibile il pacchetto xfp, curve2e è stato modificato per farne largo uso; però esiste un certo numero di utenti che non amano, o non sono interessati, o non possono caricarsi versioni più aggiornate del software del sistema TeX; i motivi possono essere svariati e giustificabili. Però aggiornare un pacchetto che si basa su funzionalità apparse verso la fine della validità di TeX Live 2018, avrebbe significato l'impossibilità di ricompilare vecchi documenti, che richiedono versioni precedenti del nucleo di LaTeX. Non è frequente, ma succede. Per questo curve2e ha un test iniziale che verifica la disponibilità di xfp e, se questo non è disponibile, carica la sua versione precedente a questo *upgrade*. Per ricompilare vecchi documenti questo è più che sufficiente. Ma se si vogliono sperimentare le macro descritte in questo articolo, è necessario servirsi di

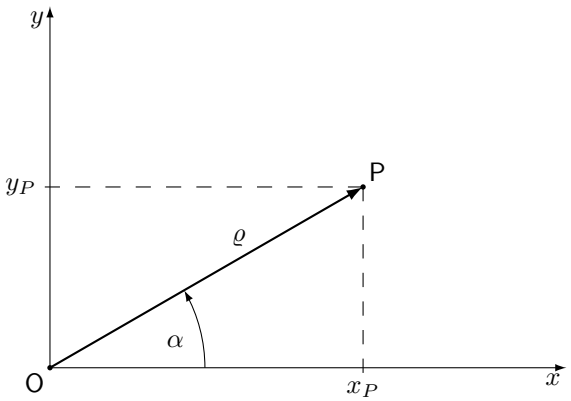

FIGURA 1: Coordinate cartesiane e polari

una versione aggiornata e completa di TEX Live o di MiKTEX.

2.2 Le estensioni attuali di curve2e

La particolarità di curve2e è che le coppie ordinate che vengono usate per indicare i punti del piano euclideo, o le direzioni in tale piano, sono trattate come numeri complessi. In realtà è solo una questione di terminologia; anche METAFONT, prima, e METAPOST, dopo, usano le coppie ordinate e le gestiscono come se fossero numeri complessi. Matematici e ingegneri hanno qualche differenza nella nomenclatura, e usano due terminologie diverse, ma per gli ingegneri i numeri complessi sono visti preferibilmente come vettori o operatori vettoriali; qui non mi riferirò alle loro terminologie, né alle proprietà di vettori, versori, operatori vettoriali, coppie ordinate, né alle classi di equipollenza e altre nomenclature relative a questi oggetti; tuttavia mi sarà comodo usare la qualifica di "complesso e coniugato" per certi numeri complessi e riferirmi all'"angolo" dei numeri complessi col nome di *argomento*.

La versione di curve2e disponibile in questo momento in cui sto scrivendo questo testo è successiva alla 2.0.8 del 2020-01-18. In quella versione, e nelle successive, tutte le macro disponibili per l'utente hanno la possibilità di scrivere le coordinate dei punti e le direzioni di segmenti e vettori indifferentemente mediante le coordinate cartesiane o polari (vedi la figura 1), racchiuse fra le normali parentesi tonde o senza delimitatori a seconda della sintassi dei vari comandi. La sintassi è la seguente:

$$\mathsf{P} = \begin{cases} x, y & \text{coordinate cartesiane} \\ \alpha: \varrho & \text{coordinate polari} \end{cases}$$

Inoltre, l'utente può definire macro per designare determinati punti del disegno con dei nomi che gli siano comodi, mnemonicamente legati a un particolare punto. Se se ne fa un uso corretto, le varie macro interne non interferiscono con i nomi usati dall'utente.[1] Di ogni macro che individua un punto si possono estrarre le componenti cartesiane o le componenti polari o trasformare le une nelle altre. Internamente, le macro di servizio usano quasi esclusivamente le coordinate cartesiane.

Per fare queste cose, le funzionalità che il pacchetto mette a disposizione, documentate nella descrizione del pacchetto xfp, sono quelle che ci si aspetta da una qualunque calcolatrice tascabile o dalle *app* del sistema operativo che abbiano la modalità *scientifica*; quindi, oltre alle quattro operazioni e alle radici, sono disponibili le funzioni circolari dirette e inverse, gli esponenziali e i logaritmi e altre utili funzioni; le funzioni circolari, con nomi diversi, accettano o i radianti o i gradi; in curve2e si accettano solo i gradi e si usano solo le funzioni trigonometriche per gestire i gradi.

Quello che non è citato nella documentazione di xfp consiste nel fatto che per il calcolo dell'arcotangente accetta sia un solo argomento, sia una coppia di valori separati da una virgola, in modo che può calcolare il valore (in radianti o) in gradi sia nell'intervallo $-90° < \alpha \leq 90°$ (se viene dato un solo argomento), o nell'intervallo $-180° < \alpha \leq 180°$, se vengono specificati due valori.[2] Lavorando con i numeri complessi, curve2e non ha più bisogno di fare lunghi test per calcolare l'argomento del numero a seconda del quadrante in cui si trova.

Dopo queste importanti modifiche, tutte le macro di curve2e per eseguire operazioni sui numeri complessi vengono eseguite correttamente con le nuove funzionalità; le macro assumono una forma facilmente decifrabile e perciò comode da curare nel caso in cui si manifestassero inconvenienti di qualche tipo. Ma anche le operazioni sulle lunghezze o in generale sui numeri reali sono molto più chiare e immediate. La divisione, in particolare, non solo si riduce a un'unica funzione del linguaggio L3, ma nemmeno richiede acrobazie per evitare gli overflow che con la versione precedente si manifestavano raramente, ma non erano esclusi. Anche la radice quadrata è formata da una sola funzione L3, e non è necessario ricorrere a nessun procedimento iterativo; prima, invece, era necessario ricorrere al procedimento di Newton.

Questo non vuol dire che ora il tempo di esecuzione sia diminuito; infatti, l'implementazione delle funzioni per il calcolo dei numeri decimali in virgola mobile e l'analisi delle espressioni da calcolare richiede non poche elaborazioni dietro le quinte, tutte codificate rigorosamente in linguaggio L3. Tuttavia, oggi gli elaboratori sono così veloci che il tempo di calcolo ammonta a pochi millisecondi, quindi generalmente si tratta di tempi del tutto trascurabili. Ciò che oggi "rallenta" l'elaborazione

1. In verità la macro `\Pi`, che mi sarebbe piaciuto usare per etichettare un "punto iniziale" non può essere usata dall'utente, perché L3 usa questa macro per indicare una certa versione del numero "pi greco".

2. Attenzione: il primo argomento riguarda l'asse verticale e il secondo l'asse orizzontale. Tuttavia, esiste anche la funzione arcocotangente che scambia di posizione i due valori, quindi basta solo essere attenti a usare la funzione giusta. Insomma, se è $P = (x, y)$, l'argomento di P si può calcolare sia con $\arctan(y/x)$, sia con $\operatorname{arccot}(x/y)$.

di un documento sono principalmente le operazioni di lettura e scrittura su disco, anche se la macchina è dotata di un disco allo stato solido. Proprio per questo, l'esecuzione di alcuni disegni eseguiti con le nuove macro talvolta può richiedere alcuni secondi, ma è quasi tutto tempo impegnato nelle operazioni di *paging*, cioè nel caricare e scaricare ripetutamente la memoria virtuale del calcolatore. Di solito questi improvvisi rallentamenti derivano da registri di memoria troppo pieni legati a oggetti flottanti di grandissime dimensioni; questo non succede solo con questo pacchetto, ma accomuna anche Ti*k*Z, tanto per citare un esempio.

Quello che il pacchetto xfp oggi non fornisce ancora, sono i comandi per gestire i test e per eseguire cicli; per questo nella nuova versione di curve2e compaiono anche le righe seguenti scritte con l'interfaccia per il linguaggio L3:

```
\ExplSyntaxOn
\AtBeginDocument{%
\ProvideExpandableDocumentCommand\fptest%
    {m m m}{\fp_compare:nTF{#1}{#2}{#3}}
\ProvideExpandableDocumentCommand\fpdowhile%
    {m m}\fp_do_while:nn{#1}{#2}
}
\ExplSyntaxOff
```

Le loro sintassi sono le seguenti:

```
\fptest{⟨test⟩}⟨vero⟩⟨falso⟩
\fpdowhile{⟨test⟩}⟨operazioni da ripetere⟩
```

Nel primo comando si esegue un ⟨*test*⟩ su espressioni relazionali (a loro volta con operandi costituiti da numeri interi o fratti e operatori di relazione come `>, <, =`) legate con gli operatori logici `||` (OR), `&&` (AND), `!` (NOT) e altri operatori meno comuni. Sono previste entrambe le uscite per eseguire il contenuto del primo o del secondo argomento che segue il test a seconda che il test risulti vero o falso.

Nel secondo comando indicato sopra si esegue un ciclo `do while`; nel primo argomento va il ⟨*test*⟩ da eseguire per controllare le iterazioni, mentre nel secondo, fra le ⟨*operazioni da ripetere*⟩ ciclicamente, deve esserci anche la modifica di qualcosa che influisca sul ⟨*test*⟩, affinché esso diventi prima o poi falso ed eviti un ciclo infinito. È evidente che prima di attivare questa funzione L3, bisogna impostare i parametri da cui dipende il ⟨*test*⟩ in modo che questo sia inizialmente vero.

I comandi che più si sono giovati delle nuove funzionalità sono quelli per le operazioni ripetitive come `\multiput` e il nuovo `\xmultiput`. Il primo è perfettamente retrocompatibile con la versione originale, ma la sua definizione consente di fare cose impossibili con quella versione. La sintassi è

```
\multiput[⟨shift⟩](⟨inizio⟩)(⟨incremento⟩)
    {⟨numero⟩}{⟨oggetto⟩}[⟨operazioni⟩]
```

Infatti tolti il primo e l'ultimo argomento facoltativi, la sintassi è identica a quella del comando originale; lo ⟨*shift*⟩ è una coppia ordinata di valori che serve per spostare il disegno prodotto da `\multiput` di quanto specificato con quel numero complesso. Le ⟨*operazioni*⟩ permettono di modificare l'incremento del valore specificato alle funzioni L3 che si possono usare in questo argomento. Il comando `\xmultiput` ha una sintassi del tutto simile, ma il controllo delle iterazioni è eseguito in modo diverso e più adatto per certi scopi; si veda per esempio la figura 5 nella pagina 9.

Le preesistenti macro dell'ambiente curve2e `\Dashline` e `\Dotline` sono ridefinite usando queste nuove modalità di calcolo, e certi inconvenienti che prima si manifestavano non si manifestano più.

Nelle figure 2 e 3 si vedono dei fasci di linee tratteggiate e punteggiate descritte con le direzioni espresse in forma sia cartesiana sia polare.

Il comando `\multiput` si avvale di un contatore interno definito con i nomi di TEX, mentre i nomi dei contatori di LATEX sono preceduti dalla stringa `c@` prima di poter accedere ai contatori TEX. La macro `\multiput` contiene al proprio una ridefinizione del contatore in modo da renderlo accessibile alle ⟨*operazioni*⟩ attraverso il nome `multicnt` mediante i comuni comandi LATEX come, per esempio, `\value`.[3]

Nella figura 4 appaiono diversi esempi di uso del comando `\multiput`.

Si noti che le macro `\R` e `\D` sono macro interne a `\multiput` e indicano rispettivamente la posizione incrementale e l'incremento ciclico del processo iterativo. Nei primi due esempi si è usato il comando `\multiput` nel modo tradizionale; nel terzo esempio si è usato anche il primo argomento facoltativo del nuovo comando `\multiput` per dislocare lo stesso codice del primo esempio di 5 unità in orizzontale e altrettanto in verticale; si è usato un quadratino invece di un dischetto in modo da mostrare la differenza di posizione.

Nel quarto esempio si è usato un incremento in coordinate polari; esso contiene un angolo di rotazione di 30° ma ha un modulo unitario; quindi può essere usato come operatore di rotazione; le stelline sono quindi collocate lungo un cerchio a distanza di 30° l'una dall'altra, ma per ottenere questo effetto si è usato il secondo argomento facoltativo dove la posizione `\R` viene ruotata moltiplicandola per il numero complesso di rotazione `\D`, in modo che il successivo ⟨*oggetto*⟩, la stellina, sia collocata lungo il cerchio. Se l'incremento non avesse avuto il modulo unitario, le stelline sarebbero state collocate lungo un arco di spirale.

3. Bisogna ricordarsi però di *non* usare i comandi LATEX `\setcounter`, `\addtocounter` e simili, perché sono comandi globali, cioè, sono comandi il cui effetto non è soggetto al limite del gruppo, o dell'ambiente, o della scatola e influisce sull'intero documento.

```
\begin{picture}(40,30)
\AutoGrid
\Dashline(0,0)(40,10){2}\Dashline(0,0)(40,20){2}
\Dashline(0,0)(40,30){2}\Dashline(0,0)(30,30){2}
\Dashline(0,0)(20,30){2}\Dashline(0,0)(10,30){2}
\Dashline(40,0)(108:30){2}
\Dashline(40,0)(126:30){2}
\Dashline(40,0)(144:30){2}
\Dashline(40,0)(162:30){2}
\end{picture}
```

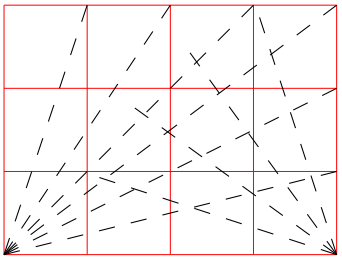

FIGURA 2: Alcune linee tratteggiate con l'⟨*incremento*⟩ espresso in forma cartesiana o polare

```
\begin{picture}(40,30)
\AutoGrid
\Dotline(0,0)(40,10){1.5}[2]\Dotline(0,0)(40,20){1.5}[2]
\Dotline(0,0)(40,30){1.5}[2]\Dotline(0,0)(30,30){1.5}[2]
\Dotline(0,0)(20,30){1.5}[2]\Dotline(0,0)(10,30){1.5}[2]
\Dotline(40,0)(108:30){1.5}[2]
\Dotline(40,0)(126:30){1.5}[2]
\Dotline(40,0)(144:30){1.5}[2]
\Dotline(40,0)(162:30){1.5}[2]
\end{picture}
```

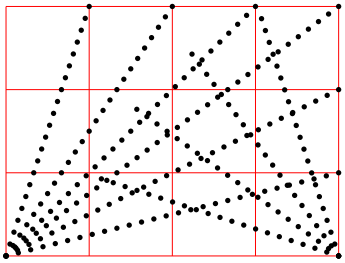

FIGURA 3: Alcune linee punteggiate con l'incremento espresso in forma cartesiana o polare

```
\unitlength=0.01\textwidth
\begin{picture}(100,100)
\AutoGrid
\multiput(0,0)(10,10){10}{\circle*{1.5}}
\multiput(0,10)(10,10){10}{\circle*{1.5}}
\multiput[4,4](0,0)(10,10){10}
   {\polygon*(0,0)(2,0)(2,2)(0,2)}
\multiput[50,50](0,30)(30:1){12}%
   {\makebox(0,0){\color{blue}\Huge$\star$}}
   [\MultVect\R by\D to\R]%
\multiput[50,50](90:36)(30:1){12}%
   {\makebox(0,0){\Roman{multicnt}}}%
   [\Multvect{\R}{\D}{\R}]
\end{picture}
```

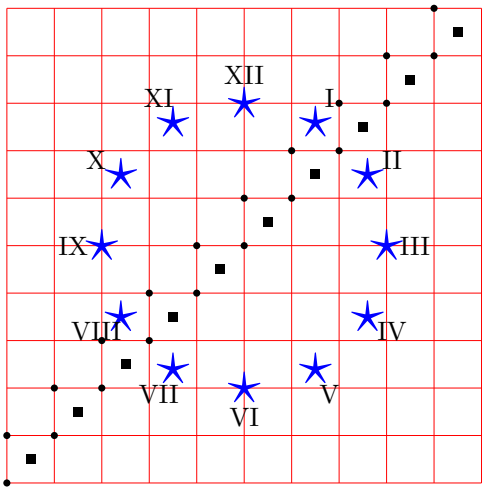

FIGURA 4: Alcuni esempi d'uso del comando `\multiput`

Il quinto esempio è del tutto simile al quarto per quel che riguarda la posizione dell'oggetto, che però questa volta è il contatore di iterazione contenuto nel contatore TEX `\multicnt` camuffato da contatore LATEX, così da potergli applicare la trasformazione in numeri romani maiuscoli attraverso il comando `\Roman`.

Sono esempi banali, tuttavia mostrano le nuove funzionalità del comando `\multiput`.

Il comando `\xmultiput` differisce dal comando standard nel fatto che il ciclo iterativo è eseguito e controllato dalla funzione L3 `\fpdowhile` definita precedentemente. L'utente è più libero nel controllare la posizione dell'oggetto e, volendo, può anche usare i comandi di disegno di basso livello come `\moveto`, `\lineto` e soci, per disegnare poligoni regolari e/o curve continue a tratti; lavorando con punti sufficientemente vicini, si possono disegnare anche curve mediante spezzate che sembrano delle curve "dolci"; certo, i punti devono essere piuttosto vicini per ottenere un effetto gradevole. Nella figura 5 è mostrato il codice e il risultato di questa operazione per disegnare un arco di iperbole.

Il trucco di definire per nome un registro numerico, qui identificato con il numero 2560 [4], rende le varie operazioni molto più semplici da leggere; il contatore identificato con `\I`, usciti dall'ambiente picture, non è più accessibile con quel nome e, se precedentemente conteneva un valore, esso viene ripristinato col valore precedente.

Questo modo di usare i nuovi comandi in linguaggio L3, rende anche agevole disegnare curve

4. Con le versioni moderne del sistema TEX non è vietato usare numeri superiori a 255.

```
\unitlength =0.008\linewidth
\begin{picture}(100,100)
\AutoGrid
\VECTOR(0,0)(100,0)\Pbox(100,0)[tr]{x}[0]
\VECTOR(0,0)(0,100)\Pbox(0,100)[tr]{y}[0]
\Pbox(0,0)[r]{O}[3pt]
\thicklines
\moveto(10,100)\countdef\I=2560 \I=11
\xmultiput(0,0)(1,0){101}%
  {\lineto(\I,\fpeval{1000/\I})}%
  [\advance\I by1 \value{multicnt}\I]
\strokepath
\end{picture}
```

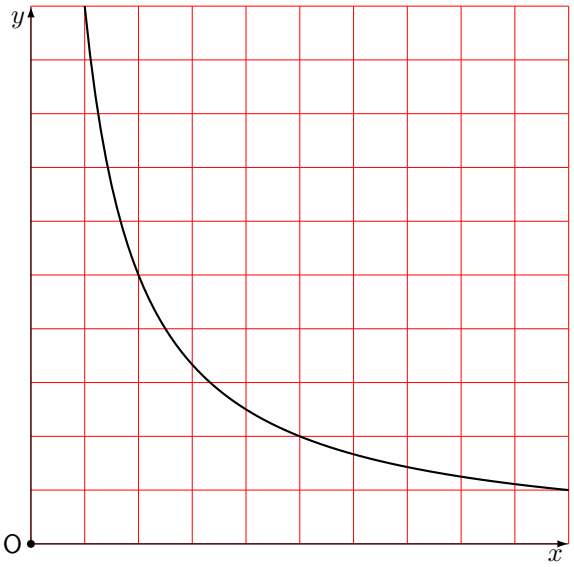

Figura 5: Disegno di un'iperbole mediante una spezzata

definite con equazioni parametriche. A titolo di esempio, si traccia una curva a forma di cuore[5], le cui equazioni parametriche sono le seguenti[6]:

$$x(t) = \sin^3(t)$$
$$y(t) = \frac{13\cos(t) - 5\cos(2t) - 2\cos(3t) - \cos(4t)}{16}$$

Conviene definire una macro tale che, ricevuto in ingresso il valore del parametro, produca in uscita il numero complesso $P = (x, y)$; bisogna anche che tenga conto delle scale del disegno, mediante un coefficiente di scala `\scala`; conviene che l'intero disegno sia centrato anche verticalmente.

```
\newcommand\cuore[3]{%
\edef\X{\fpeval{#1*16*(sind(#2)^3)}}
\edef\Y{\fpeval{#1*(13*cosd(#2) - 5*cosd(2*#2)
  - 2*cosd(3*#2) -cosd(4*#2)+2.4)}}
\CopyVect\X,\Y to#3}
```

In questa macro il primo argomento riceve il fattore di scala, il secondo il valore del parametro e il terzo produce in uscita una macro che contiene il numero complesso calcolato: la costante 2.4 che compare nel codice serve per spostare il disegno in alto in modo che sia centrato.

Ciò fatto, il disegno del diagramma e il suo codice sono mostrati nella figura 6.

5. Quanto qui esposto è una possibile soluzione ad un quesito posto sul forum del GuIT. In quel quesito si indicava il sito https://tex.stackexchange.com/questions/139733/can-we-make-a-love-heart-with-latex, dove sono state date molte soluzioni, una più bella dell'altra, ma nessuna eseguita con curve2e. Nella risposta che ho dato nel forum, ho mostrato una soluzione ottenuta con la macro `\Curve` (con o senza asterisco) che però è un disegno eseguito con spline di Bézier, non il tracciamento della curva parametrica come fatto qui.

6. Vedi http://mathworld.wolfram.com/HeartCurve.html, che contiene le equazioni di diversi "cuori" di forme diverse

Si noti che in questo caso il parametro indica il valore dell'angolo in gradi; infatti, lavorando con i numeri interi si è sicuri di far variare il parametro esattamente da 0° a 360°; inoltre il passo di incremento del parametro viene fissato a 3°; l'esecuzione del disegno avviene in modo decisamente più rapido rispetto al tempo che si avrebbe scegliendo un valore più piccolo, per esempio 1°; nello stesso tempo la figura non mostra le giunzioni fra i brevi segmenti che formano la spezzata. Infine, se al posto di `\strokepath`, si usa `\fillpath`, quel contorno viene riempito con il colore corrente; l'impostazione del colore va fatta prima di attivare il comando `\fpdowhile`.

2.3 Il comando \Pbox

I vari esempi mostrati fino a questo punto mostrano l'uso di alcune delle funzionalità presenti nell'ultima versione del pacchetto curve2e. Il lettore può vedere qui come sia definito quel comando `\Pbox` che serve per mettere l'etichetta agli assi e agli elementi geometrici che compaiono nelle figure:

```
\providecommand\Pbox{}
\RenewDocumentCommand\Pbox
  {D(){0,0} O{cc} m O{0.5ex}}{%
\put(#1){%
\dimendef\Dim=2566\relax
\settowidth\Dim{#2}%
\edef\Rapp{\fpeval{\Dim/{1ex}}}%
\fptest{\Rapp > 1.5}%
{\fboxsep=0.5ex}{\fboxsep=0.75ex}%
\fboxrule=0pt
\fptest{#4 = 0sp}%
  {\makebox(0,0)[#2]{\fbox{$\relax#3\relax$}}}%
  {\edef\Diam{\fpeval{#4/\unitlength}}%
   \makebox(0,0){\circle*{\Diam}}%
   \makebox(0,0)[#2]{%
     \fbox{$\relax\mathsf#3\relax$}}}%
}\ignorespaces}
```

La sua versatile sintassi è la seguente:

```
\unitlength=0,004\linewidth
\begin{picture}(200,200)(-100,-100)
\AutoGrid
\VECTOR(-100,0)(100,0)\Pbox(100,0)[br]{x}[0]
\VECTOR(0,-100)(0,100)\Pbox(0,100)[tl]{y}[0]
\Pbox(0,0)[tr]{O}\linethickness{1pt}\bgroup
\edef\scala{\fpeval{100/16}}
\countdef\I=2560 \I=0\roundjoin
\fpdowhile{\I !>360}{\cuore\scala\I\Punto
\ifnum\I=0 \moveto(\Punto)\else \lineto(\Punto)\fi
\advance\I by 3}\strokepath\egroup
\end{picture}
```

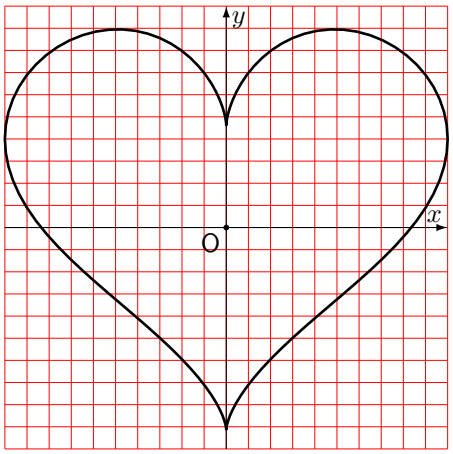

Figura 6: Disegno di una curva a forma di cuore

`\Pbox(`⟨*posizione*⟩`)[`⟨*allineamento*⟩`]%`
`{`⟨*etichetta*⟩`}[`⟨*diametro*⟩`]`

dove la ⟨*posizione*⟩ indica dove verrà collocato il pallino che identifica l'oggetto da etichettare; se il pallino ha un ⟨*diametro*⟩ nullo, è invisibile; in ogni caso, il ⟨*diametro*⟩ viene specificato con unità di misura esplicite[7] e ci pensa la macro a trasformarle in multipli di `\unitlength`; l'⟨*etichetta*⟩ viene sempre composta in modo matematico, ma se il ⟨*diametro*⟩ è non nullo, essa identifica un oggetto non misurabile e, secondo le norme ISO, viene usato il carattere senza grazie; se la si vuole in modo testo, bisogna racchiuderla esplicitamente fra due segni di dollaro, per passare dal modo matematico predefinito al modo testo; l'⟨*allineamento*⟩, formato dai soliti codici `c` (*center*), `t` (*top*), `b` (*bottom*), `r` (*right*), `l` (*left*), specifica (di fatto) la *posizione* del pallino rispetto all'etichetta; il doppio allineamento centrato (verticale e orizzontale) predefinito, serve per produrre un risultato inaccettabile, che segnala all'utente l'obbligo di specificare i propri codici di posizionamento.

`tl`	`t`	`tr`	`r`	`l`	`bl`	`b`	`br`
NW	N	NE	E	W	SW	S	SE

I punti cardinali indicati sotto i codici forse sono più espliciti per indicare la ⟨*posizione*⟩ del pallino rispetto all'etichetta.

3 Poligoni regolari, triangoli ed ellissi

Le macro e le estensioni descritte nel paragrafo precedente tornano utili per disegnare curve di vario genere e per eseguire le costruzioni con riga, squadra e compasso tipiche della geometria euclidea.

7. Volendo si può omettere l'unità di misura, nel qual caso il valore è assunto come misurato in punti tipografici.

3.1 Poligoni regolari

Per disegnare poligoni regolari col solo contorno o ripieni di colore, variamente orientati, con i lati di spessore diverso da quello predefinito, si può definire una macro che faccia tutto in un colpo solo. Eccone il codice.

```
\NewDocumentCommand\RegPolygon%
  {s D(){0,0}  m m O{0} D<>{\relax} }{{%
\countdef\I=258 \I=0
\CopyVect#5:#3to\P
\CopyVect\fpeval{360/#4}:1to\R
\put(#2){#6\relax
  \moveto(\P)\fpdowhile{\I < #4}%
  {\MultVect\P by\R to\P
  \lineto(\P)\advance\I by 1}%
  \IfBooleanTF{#1}%
  {\fillpath}{#6\strokepath}}}\ignorespaces}
```

La sintassi della macro è la seguente:

`\RegPolygon`⟨⋆⟩`(`⟨*centro*⟩`){`⟨*raggio*⟩`}%`
`{`⟨*numero*⟩`}[`⟨*angolo*⟩`]<`⟨*impostazioni*⟩`>`

Il primo asterisco è facoltativo; se c'è, l'area racchiusa dal poligono viene colorata con il colore corrente, oppure vengono usati il colore e/o gli spessori specificati con le ⟨*impostazioni*⟩; si possono specificare le coordinate del ⟨*centro*⟩, ma se non lo si fa, il centro viene messo nell'origine degli assi. Bisogna evidentemente specificare il ⟨*raggio*⟩ del cerchio circoscritto e il ⟨*numero*⟩ di lati; facoltativamente si può specificare (in gradi) l'angolo di rotazione del primo vertice; infine, racchiuse fra i segni `< >`, si possono indicare le ⟨*impostazioni*⟩ che si desiderano. Se si vuole il perimetro del poligono disegnato con un colore diverso da quello del suo interno, bisogna sovrapporre il poligono col solo contorno al poligono colorato internamente. La figura 7 contiene diversi esempi; essa è ottenuta con il codice seguente:

```
\unitlength=0.006\linewidth
\begin{picture}(120,100)
```

```
%
\RegPolygon(9,20){20}{6}%
  <\linethickness{3pt}\color{red}>
\RegPolygon(55,20){20}{7}[90]
\RegPolygon(100,20){20}{8}[22.5]%
  <\linethickness{0.5ex}\color{blue}>
\put(0,50){%
\RegPolygon(20,20){20}{3}
\RegPolygon(20,20){20}{3}[30]
\RegPolygon(20,20){20}{3}[60]
\RegPolygon(20,20){20}{3}[90]
%
\RegPolygon*(62,20){20}{4}<\color{green}>
\RegPolygon(62,20){20}{4}%
  <\linethickness{1ex}>
%
\RegPolygon*(100,20){20}{4}[45]%
  <\color{orange}>
\RegPolygon(100,20){20}{4}[45]%
  <\linethickness{1ex}\color{blue}>
}
\end{picture}
```

3.2 Ellissi

Il pacchetto curve2e non contiene macro per disegnare ellissi. In realtà, il loro disegno di base è semplicissimo; il codice è il seguente:

```
\def\ellisse#1#2{%
\bgroup\edef\ELa{#1}\edef\ELb{#2}%
\edef\ELx{\fpeval{4*(sqrt(2)-1)/3}}%
\edef\ELax{\fpeval{\ELx*\ELa}}%
\edef\ELbx{\fpeval{\ELx*\ELb}}%
\moveto(\ELa,0)
\curveto(\ELa,\ELbx)(\ELax,\ELb)(0,\ELb)
\curveto(-\ELax,\ELb)(-\ELa,\ELbx)(-\ELa,0)
\curveto(-\ELa,-\ELbx)(-\ELax,-\ELb)(0,-\ELb)
\curveto(\ELax,-\ELb)(\ELa,-\ELbx)(\ELa,0)
\fillstroke\egroup\ignorespaces}
```

Questo codice funziona benissimo, ma non è sufficientemente versatile; in particolare, non lo è per gli scopi di questo articolo. È solo usato internamente da `\Xellisse`, che ne imposta tutte le caratteristiche e, in particolare, specifica il significato della macro `\fillstroke`; questa viene in effetti posta equivalente a `\strokepath` quando si vuole disegnare solo il contorno, oppure a `\fillpath` quando si vuole riempire il contorno con un colore. La macro per l'utente è la seguente:

```
\NewDocumentCommand{\Xellisse}%
  { s D(){0,0} O{0} m m O{} o}%
    {\IfBooleanTF{#1}%
      {\let\fillstroke\fillpath}%
      {\let\fillstroke\strokepath}%
    \put(#2){\rotatebox{#3}{{#6\ellisse{#4}{#5}}%
    \IfValueTF{#7}{\let\fillstroke\strokepath
    #7\ellisse{#4}{#5}}{}}}%
  }
```

I suoi sette argomenti permettono di fare qualunque cosa con l'ellisse; sia per usare i colori sia per disegnarne il contorno; a differenza dal comando `\RegPolygon`, non richiede all'utente di disegnare due ellissi per disegnarne il contorno sovrapposto al solo interno colorato, perché fa tutto da solo; anzi, questo codice potrebbe essere preso a modello per ridefinire `\RegPolygon` perché si comporti nello stesso modo. La sua sintassi è la seguente:

`\Xellisse`⟨⋆⟩`(`⟨*centro*⟩`)[`⟨*angolo*⟩`]%`
`{`⟨*semiassse-a*⟩`}{`⟨*semiasse-b*⟩`}%`
`[`⟨*impostazioni-1*⟩`][`⟨*impostazioni-2*⟩`]`

L'asterisco facoltativo controlla il riempimento di colore; se manca, viene disegnato solo il contorno e le ⟨*impostazioni-2*⟩, che normalmente ricevono le impostazioni per il contorno, possono essere omesse in quanto collocabili anche nel campo ⟨*impostazioni-1*⟩; viceversa, se l'asterisco è presente, le ⟨*impostazioni-1*⟩ riguardano il riempimento, mentre le ⟨*impostazioni-2*⟩, facoltative, riguardano il contorno. Per l'ellisse si deve esplicitare il ⟨*centro*⟩, si può specificare l'⟨*angolo*⟩ di rotazione dell'intera ellisse; ovviamente si devono specificare il semiasse principale (maggiore) ⟨*semiasse-a*⟩ e quello secondario (minore) ⟨*semiasse-b*⟩, anche se, in realtà, la macro non fa differenze fra i due semiassi: il primo è quello che, senza rotazione, è orizzontale e il secondo quello verticale. Le impostazioni sono già state descritte prima. Si fa notare che queste impostazioni di colore per il riempimento e il bordo sono utili in altri contesti, non solo nelle costruzioni geometriche; l'uso è già stato descritto in un altro articolo (Beccari, 2018a). Qui quello che interessa è l'oggetto geometrico "ellisse", più che le sue decorazioni; tuttavia, anche nelle applicazioni di questo articolo, non è fuori luogo impostare lo spessore del contorno.

Per tornare a questo articolo è meglio usare una macro un po' più complessa, ma che permetta di operare trasformazioni affini sull'ellisse; fra queste compaiono gli spostamenti, le rotazioni e gli scorrimenti, *shear* in inglese[8]. Lo scorrimento di una griglia di rette incrociate ad angolo retto, diventa una griglia di rette che si incrociano con un angolo diverso; per esemplificare con una figura geometrica comune, un rettangolo diventa un parallelogramma con angoli interni diversi da 90°. Non ci sono problemi con gli spostamenti, perché basta specificare le coordinate del ⟨*centro*⟩; non ci sono problemi con le rotazioni, perché si può sempre specificare l'⟨*angolo*⟩. Invece, per le trasformazioni affini, il nucleo di LaTeX e il pacchetto curve2e non dispongono di nessun comando. Il pacchetto xpicture (Fuster, 2012), invece, le gestisce; infatti è pensato proprio per gli scopi di un matematico, ma forse è un po' troppo complesso da usare quando il

8. Talvolta si usa la parola *skew*, ma questo ha un generico significato di inclinazione, mentre *shear* si riferisce ad un preciso scorrimento laterale "a strati".

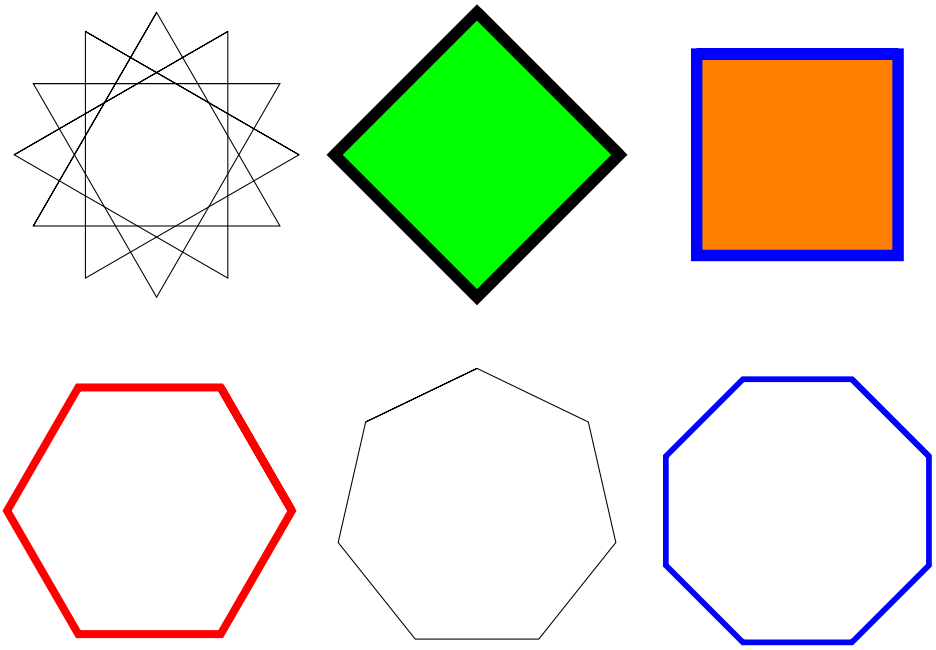

Figura 7: Alcuni poligoni

problema si può risolvere con una macro; vedremo più avanti come.

La figura 8 contiene alcuni esempi ottenuti con il codice seguente:

```
\unitlength=0.007\linewidth
\begin{picture}(100,100)
\AutoGrid
\Xellisse(30,20){30}{20}
  \Pbox(30,20)[l]{C_1}[2pt]
\Xellisse(80,50)[90]{30}{20}
  \Pbox(80,50)[l]{C_2}[2pt]
\Xellisse(30,70)[45]{30}{20}%
  [\linethickness{1pt}]
  \Pbox(30,70)[l]{C_3}[2pt]
\end{picture}
```

Si vedrà più avanti come sostituire la macro `\ellisse` con un'altra che consenta di eseguire anche la trasformazione affine di scorrimento.

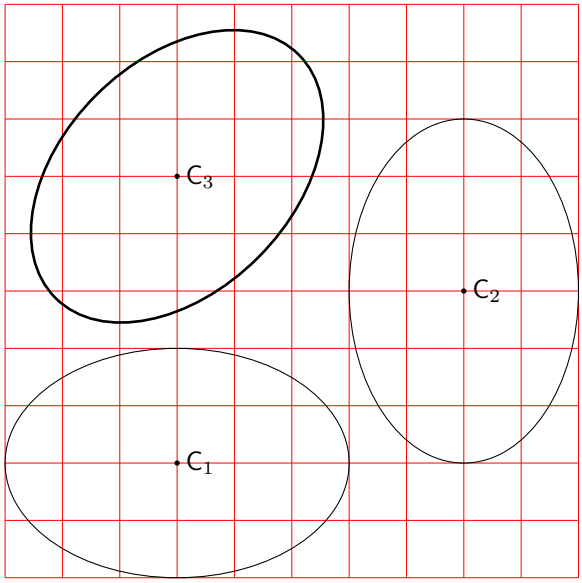

Figura 8: Alcune ellissi

3.3 Triangoli

È chiaro che un triangolo è il più semplice dei poligoni, ma questa figura ha diverse proprietà, in particolare dispone di diversi centri: il baricentro, l'ortocentro, l'incentro, il circocentro, il centro del cerchio dei nove punti, eccetera. Qui mostreremo come determinare questi centri ed eventualmente come disegnare i cerchi associati ad alcuni di questi centri.

Intanto, ecco un disegno con un triangolo e alcune sue linee importanti: mediana, bisettrice e altezza, figura 9. Le macro che servono per tracciare queste linee o i loro estremi sono esposte nei paragrafi successivi.

3.3.1 Il baricentro

Il *baricentro* è il punto di intersezione delle mediane, cioè i segmenti che uniscono ogni vertice con il punto mediano del lato opposto; vedi la figura 10.

Il codice per disegnare la figura è il seguente:

```
\unitlength=0.01\linewidth
\begin{picture}(100,100)
\AutoGrid
\CopyVect10,20to\Pu \CopyVect60,10to\Pd
\CopyVect80,90to\Pt
\polygon(\Pu)(\Pd)(\Pt)
\Pbox(\Pu)[tr]{P_1}[2pt] \Pbox(\Pd)[l]{P_2}[2pt]
\Pbox(\Pt)[tl]{P_3}[2pt]
\TriangleMedianBase\Pu on\Pd and\Pt to\Pmu
```

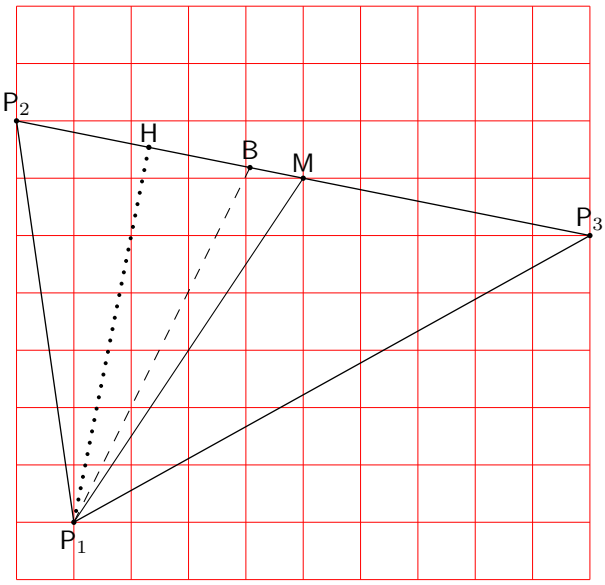

FIGURA 9: Un triangolo con una mediana, un'altezza e una bisettrice

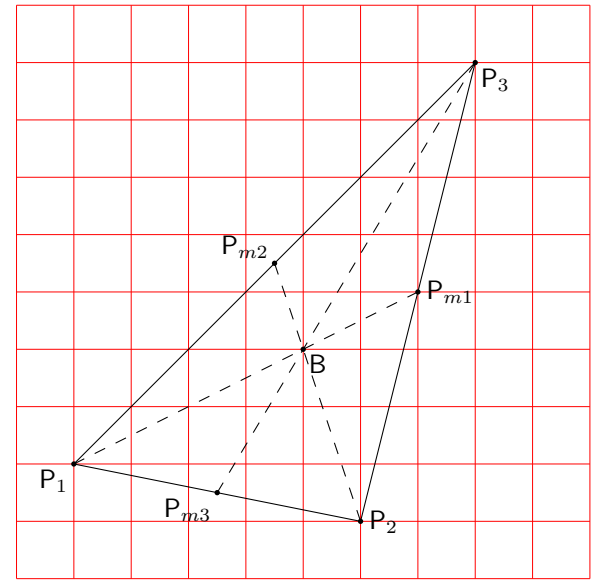

FIGURA 10: Determinazione del baricentro

```
\TriangleMedianBase\Pd on \Pt and\Pu to\Pmd
\Pbox(\Pmu)[l]{P_{m1}}[2pt]
\Pbox(\Pmd)[br]{P_{m2}}[2pt]
\Dashline(\Pu)(\Pmu){2}\Dashline(\Pd)(\Pmd){2}
\IntersectionOfSegments(\Pu)(\Pmu)%
  and(\Pd)(\Pmd)to\B
\Pbox(\B)[tl]{B}[2pt]
\end{picture}
```

Come si vede, si sono usati alcuni nuovi comandi: `\IntersectionOfSegments` e `\TriangleMedianBase`; quest'ultimo in realtà si basa a propria volta su `\IntersectionOfLines`. In realtà si sarebbe potuto determinare direttamente il baricentro con l'unica macro `\TriangleBarycenter`, ma non sarebbe stata disegnata l'intera costruzione che porta al risultato. Questi comandi sono definiti cosi:

```
\def\TriangleMedianBase#1on#2and#3to#4{%
\SubVect#1from#2to\TMBu
\SubVect#1from#3to\TMBd
\SubVect\TMBu from\TMBd to\Base
\ScaleVect\Base by0.5to\TMBm
\AddVect#2and\TMBm to#4\ignorespaces}

\def\IntersectionOfSegments(#1)(#2)and(#3)(#4)to#5{%
\SubVect#1from#2to\IoSvectu
\DirOfVect\IoSvectu to\DirIoSVecu
\SubVect#3from#4to\IoSvectd
\DirOfVect\IoSvectd to\DirIoSVecd
\IntersectionOfLines(#1)(\DirIoSVecu)%
  and(#3)(\DirIoSVecd)to#5 \ignorespaces}

\def\IntersectionOfLines(#1)(#2)and(#3)(#4)to#5{%
\bgroup
\def\IntPu{#1}\def\Uu{#2}
\def\IntPd{#3}\def\Ud{#4}%
 \DirOfVect\Uu to\Du
 \DirOfVect\Ud to\Dd
 \XpartOfVect\Du to \a
 \YpartOfVect\Du to \b
 \XpartOfVect\Dd to \c
 \YpartOfVect\Dd to \d
 \XpartOfVect\IntPu to \xu
 \YpartOfVect\IntPu to \yu
 \XpartOfVect\IntPd to \xd
 \YpartOfVect\IntPd to \yd
 \edef\Den{\fpeval{-(\a*\d-\b*\c)}}%
 \fptest{abs(\Den)<1e-5}%
 {% determinante quasi nullo
   \def#5{0,0}%
 }{% Determinante non quasi nullo
   \edef\Numx{\fpeval{(\c*(\b*\xu-\a*\yu)
     -\a*(\d*\xd-\c*\yd))/\Den}}%
   \edef\Numy{\fpeval{(\d*(\b*\xu-\a*\yu)
     -\b*(\d*\xd-\c*\yd))/\Den}}%
   \CopyVect\Numx,\Numy to\Paux
   \edef\x{\egroup\noexpand
     \edef\noexpand#5{\Paux}}\x\ignorespaces
 }}

\def\TriangleBarycenter(#1)(#2)(#3)to#4{%
\TriangleMedianBase#1on#2and#3to\Pa
\TriangleMedianBase#2on#3and#1to\Pb
\DistanceAndDirOfVect#1minus\Pa to\ModPa and\AngPa
\DistanceAndDirOfVect#2minus\Pb to\ModPb and\AngPb
\IntersectionOfLines(#1)(\AngPa)and(#2)(\AngPb)to#4}
```

Come si vede, il cuore di tutta l'elaborazione è costituito dalla macro `\IntersectionOfLines` basata sulla matematica seguente. Si hanno due linee ciascuna definita da un punto P_i e una direzione $u_i = \exp(\mathrm{i}\beta_i)$; queste, se non sono parallele o anti-parallele, si intersecano in un punto Q:

$$\begin{cases} Q = P_1 + t_1 u_1 \\ Q = P_2 + t_2 u_2 \end{cases}$$

Eliminando Q fra le due equazioni, e ponendo

$$P_2 - P_1 = P_D = D\,\mathrm{e}^{\mathrm{i}\delta}$$

si ottiene

$$t_1 u_1 - t_2 u_2 = P_D$$

Infine, separando le componenti orizzontali da quelle verticali, si ha:

$$\begin{cases} t_1 \cos\beta_1 - t2\cos\beta_2 = D\cos\delta \\ t_1 \sin\beta_1 - t2\sin\beta_2 = D\sin\delta \end{cases}$$

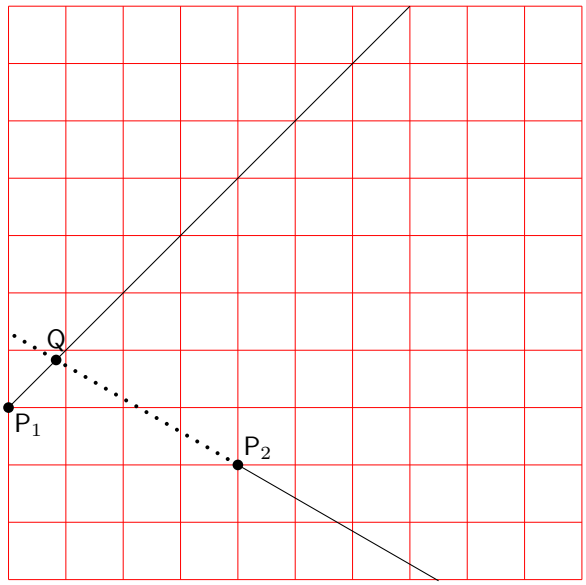

FIGURA 11: Intersezione di due rette

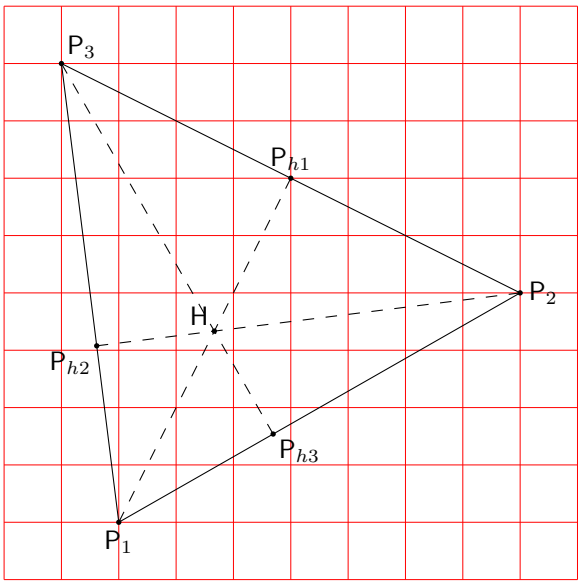

FIGURA 12: Determinazione dell'ortocentro

Risolvendo rispetto a uno dei due parametri t_i si ha, per esempio:

$$t_1 = D\frac{\sin(\delta-\beta_2)}{\sin(\beta_1-\beta_2)}$$

da cui il punto di intersezione è dato da:

$$Q = P_1 + t_1u_1 = P_1 + D\frac{\sin(\delta-\beta_2)}{\sin(\beta_1-\beta_2)}u_1$$

La figura 11 mostra la semplice costruzione geometrica. Vi si vede, fra l'altro, che l'intersezione Q avviene sul lato "negativo" della seconda retta, il cui versore punta verso destra in basso: infatti l'angolo della sua direzione `\Udue` vale $-30°$.

3.3.2 L'ortocentro

L'*ortocentro* di un triangolo è il punto dove si intersecano le tre altezze rispetto ai tre lati; in un triangolo acutangolo esso cade all'interno del triangolo; in un triangolo rettangolo coincide con il vertice dell'angolo retto; in un triangolo scaleno l'ortocentro si trova all'esterno del triangolo. La figura 12 mostra il caso di un triangolo acutangolo.

```
\unitlength=0.01\linewidth
\begin{picture}(100,100)
\AutoGrid
\CopyVect20,10to\Pu\Pbox(\Pu)[t]{P_1}[2pt]
\CopyVect90,50 to\Pd\Pbox(\Pd)[l]{P_2}[2pt]
\CopyVect10,90 to\Pt\Pbox(\Pt)[bl]{P_3}[2pt]
\polygon(\Pu)(\Pd)(\Pt)
\TriangleHeightBase\Pu on\Pd and\Pt to\Phu
\TriangleHeightBase\Pd on\Pt and\Pu to\Phd
\TriangleHeightBase\Pt on\Pu and\Pd to\Pht
\Dashline(\Pu)(\Phu){2}
\Pbox(\Phu)[b]{P_{h1}}[2pt]
\Dashline(\Pd)(\Phd){2}
\Pbox(\Phd)[tr]{P_{h2}}[2pt]
\Dashline(\Pt)(\Pht){2}
\Pbox(\Pht)[tl]{P_{h3}}[2pt]
\IntersectionOfSegments(\Pu)(\Phu)
  and(\Pd)(\Phd)to\H
\Pbox(\H)[br]{H}[2pt]
\end{picture}
```

Si noti che, come nel caso del baricentro, non si è usato il comando `\IntersectionOfLines` ma `\IntersectionOfSegments`. Non sono proprio equivalenti, perché nel primo caso bisogna specificare un punto e una direzione, mentre nel secondo caso bisogna specificare due punti (distinti). La differenza sostanziale è che, appunto, il comando per i segmenti calcola lui stesso la direzione in base agli estremi del segmento e poi invoca il comando per le linee. La direzione non è sempre intuitiva, quindi è sempre meglio lasciare fare i calcoli alle macro correttamente definite. In questo caso, invece, si è invocato il comando `\TriangleHeightBase`, che permette di tracciare anche le linee tratteggiate per eseguire la costruzione geometrica. Esiste anche il comando `\TriangleOrthoCenter`, che però determina direttamente le coordinate dell'ortocentro `\H`, ma non i punti che permettono di disegnare le linee tratteggiate. In ogni caso, quei comandi sono definiti come segue:

```
\def\TriangleHeightBase#1on#2and#3to#4{%
% #1 := Vertice del triangolo
% #2 e #3 :=   estremi del lato opposto
% #4 := piede dell'altezza
\SubVect#2from#3to\Base
\ArgOfVect\Base to\Ang
\CopyVect\fpeval{\Ang+90}:1 to\Perp
\IntersectionOfLines(#1)(\Perp)
  and(#2)(\Base)to#4\ignorespaces}

\def\TriangleOrthocenter(#1)(#2)(#3)to#4{%
\TriangleHeightBase#1on#2and#3to\Pa
\TriangleHeightBase#2on#3and#1to\Pb
\DistanceAndDirOfVect#1minus\Pa
  to\ModPa and\AngPa
```

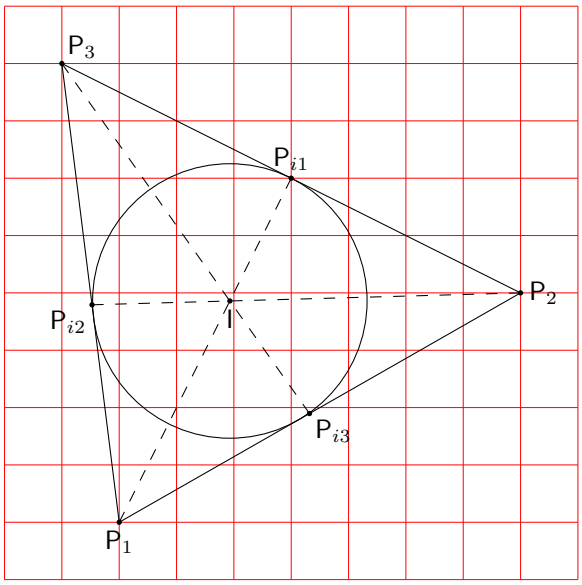

FIGURA 13: Determinazione dell'incentro e del cerchio inscritto

```
\DistanceAndDirOfVect#2minus\Pb
  to\ModPb and\AngPb
\IntersectionOfLines(#1)(\AngPa)
  and(#2)(\AngPb)to#4}
```

3.3.3 *L'incentro*

L'*incentro* è l'intersezione delle bisettrici degli angoli interni di un triangolo; ogni punto di ciascuna bisettrice è equidistante dai due lati del triangolo che racchiudono l'angolo considerato per definire la bisettrice. Il punto di incrocio delle tre bisettrici è perciò equidistante da ciascuno dei tre lati, quindi è il centro dell'incerchio[9], cioè del cerchio inscritto nel triangolo e tangente a tutti e tre i lati. È possibile disegnare l'intera costruzione dell'incentro e si può tracciare direttamente anche l'incerchio, come si vede nella figura 13.

Per disegnare la figura 13 si è usato il codice seguente dove compaiono alcune nuove macro.

```
\unitlength=0.01\linewidth
\begin{picture}(100,100)
\AutoGrid
\CopyVect20,10to\Pu\Pbox(\Pu)[t]{P_1}[2pt]
\CopyVect90,50 to\Pd\Pbox(\Pd)[l]{P_2}[2pt]
\CopyVect10,90 to\Pt\Pbox(\Pt)[bl]{P_3}[2pt]
\polygon(\Pu)(\Pd)(\Pt)
\TriangleBisectorBase\Pu on\Pd and\Pt to\Piu
\TriangleBisectorBase\Pd on\Pt and\Pu to\Pid
\TriangleBisectorBase\Pt on\Pu and\Pd to\Pit
\Dashline(\Pu)(\Piu){2}
  \Pbox(\Piu)[b]{P_{i1}}[2pt]
\Dashline(\Pd)(\Pid){2}
  \Pbox(\Pid)[tr]{P_{i2}}[2pt]
\Dashline(\Pt)(\Pit){2}
  \Pbox(\Pit)[tl]{P_{i3}}[2pt]
\IntersectionOfSegments(\Pu)(\Piu)
  and(\Pd)(\Pid)to\I
\Pbox(\I)[t]{I}[2pt]
\SegmentLength(\I)(\Piu)to\R
\Circlewithcenter\I radius\R
\end{picture}
```

Le nuove macro sono le seguenti. `\TriangleBisectorBase` che consente di disporre dei dati intermedi e di disegnare le linee tratteggiate necessarie per descrivere la costruzione. `\SegmentLength` consente di misurare la distanza di due punti che potrebbero essere gli estremi di un segmento. `\Circlewithcenter` serve per disegnare un cerchio noti il centro e il raggio (non il diametro, contrariamente alla definizione originale dell'ambiente picture) e non richiede di essere messo in posizione con `\put`. Infine `\TriangleIncenter` permette di trovare l'incentro ma non consente di usare i dati interni del codice.

```
\def\TriangleBisectorBase#1on#2and#3to#4{%
% #1 := Vertice del triangolo
% #2 e #3 :=  estremi del lato opposto
% #4 := piede della bisettrice
\SubVect#2from#1to\Luno
\SubVect#3from#1to\Ldue
\SubVect#2from#3to\Base
\ArgOfVect\Luno to\Arguno
\ArgOfVect\Ldue to\Argdue
\edef\ArgBis{\fpeval{(\Arguno+\Argdue)/2}}%
\CopyVect \ArgBis:1to \Bisect
\IntersectionOfLines(#2)(\Base)
  and(#1)(\Bisect)to#4\ignorespaces}

\def\SegmentLength(#1)(#2)to#3{%
  \SubVect#1from#2to\Segm
  \ModOfVect\Segm to#3}

\def\Circlewithcenter#1radius#2{%
\put(#1){\circle{\fpeval{2*#2}}}
\ignorespaces}

\def\TriangleIncenter(#1)(#2)(#3)to#4{%
\TriangleBisectorBase#1on#2and#3to\Pa
\TriangleBisectorBase#2on#3and#1to\Pb
\DistanceAndDirOfVect#1minus\Pa
  to\ModPa and\AngPa
\DistanceAndDirOfVect#2minus\Pb
  to\ModPb and\AngPb
\IntersectionOfLines(#1)(\AngPa)
  and(#2)(\AngPb)to#4}
```

3.3.4 *Il cerchio dei tre punti*

Un cerchio e una circonferenza sono definiti dal raggio e dalla circonferenza. Ma questi dati possono essere ricavati dalla lista di tre punti per i quali passa la circonferenza. Geometricamente è semplicissimo individuare centro e raggio: basta disegnare il triangolo che ha i tre punti come vertici,

9. La geometria distingue la *circonferenza* dal *cerchio*, essendo la prima solo il contorno del secondo. L'*incerchio* potrebbe essere chiamato *incirconferenza*, ma non si è trovato questo termine in nessun testo di geometria. In questo contesto, sembra superfluo distinguere le due cose, ma formalmente usare indifferentemente i due nomi non è corretto.

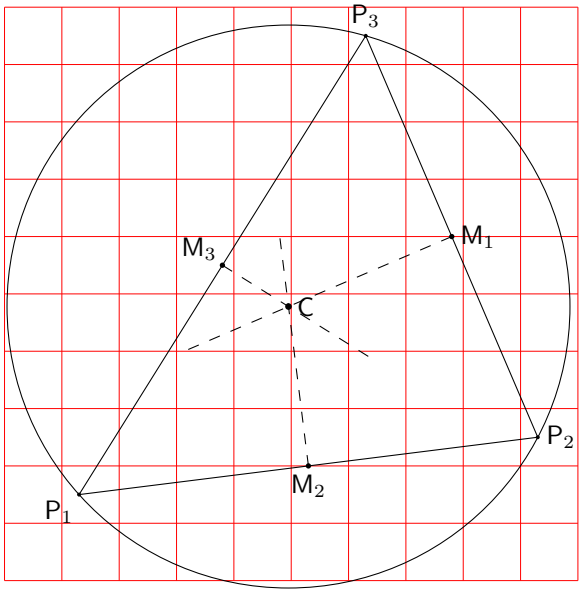

Figura 14: Circonferenza per tre punti dati

per poi incrociare gli assi di due lati. Il punto di intersezione è equidistante da tutti e tre i vertici, proprio perché si trova sugli assi di due distinti lati del triangolo. La figura 14 mostra la costruzione.

Il disegno della figura 14 è stato eseguito con il codice seguente:

```
\centering
\unitlength=0.01\linewidth
\begin{picture}(100,100)
\AutoGrid
\CopyVect13,15 to\Pu\Pbox(\Pu)[tr]{P_1}[1.5]
\CopyVect93,25 to\Pd\Pbox(\Pd)[l]{P_2}[1.5]
\CopyVect63,95 to\Pt\Pbox(\Pt)[b]{P_3}[1.5]
\polygon(\Pu)(\Pd)(\Pt)
\AxisOf\Pu and\Pd to\Md\Dd
\AxisOf\Pu and\Pt to\Mt\Dt
\AxisOf\Pd and\Pt to\Mu\Du
\IntersectionOfLines(\Md)(\Dd)and(\Mt)(\Dt)
  to\Centro
\SubVect\Md from\Centro to\Rd
\ArgOfVect\Rd to\Angd
\SubVect\Mt from\Centro to\Rt
\ArgOfVect\Rt to\Angt
\SubVect\Mu from\Centro to\Ru
\ArgOfVect\Ru to\Angu
\Dashline(\Md)(\Angd:40){2}
\Pbox(\Md)[t]{M_2}[2]
\Dashline(\Mt)(\Angt:30){2}
\Pbox(\Mt)[br]{M_3}[2]
\Dashline(\Mu)(\Angu:50){2}
\Pbox(\Mu)[l]{M_1}[2]
\ThreePointCircle*(\Pu)(\Pd)(\Pt)
\Pbox(\C)[l]{C}[2.5]
\end{picture}
```

Vi compaiono alcune nuove macro che sono riportate qui di seguito:

```
\def\AxisOf#1and#2to#3#4{%
\SubVect#1from#2to\Base
\ScaleVect\Base by0.5to\Base
\AddVect\Base and#1to#3
\MultVect\Base by0,1to#4}

\NewDocumentCommand\ThreePointCircle%
   {s d() d() d()}{%
\AxisOf#2and#3to\Mu\Du \AxisOf#2and#4to\Md\Dd
\IntersectionOfLines(\Mu)(\Du)and(\Md)(\Dd)
  to\C
\SubVect#2from\C to\R \ScaleVect\R by2to\D
\ModOfVect\D to\D
\IfBooleanTF{#1}%
  {\CircleWithCenter\C Radius\R}{}%
\ignorespaces}
```

Si noti che `\ArgOfVect` è già presente in curve2e. Nella definizione di `\ThreePointCircle` è possibile usare un asterisco facoltativo; la sua presenza permette di disegnare effettivamente il cerchio per tre punti, ma se lo si omette, rende solo accessibili le coordinate del centro dalla sua macro interna `\C`.

Senza ricorrere ad asterischi si può usare la macro `\ThreePointCircleCenter` che ha la seguente sintassi:

```
\ThreePointCircleCenter%
    (⟨P1⟩)(⟨P2⟩)(⟨P3⟩) to⟨centro⟩
```

dove ⟨*centro*⟩ è la macro che riceve le coordinate del centro del cerchio dei tre punti.

3.3.5 La circonferenza dei nove punti

Dato un triangolo, i nove punti per cui passa circonferenza sono i seguenti:

- i tre punti medi dei lati
- i piedi delle tre altezze
- i punti medi dei segmenti che uniscono ogni vertice con l'ortocentro

Questa è una proprietà geometrica dei triangoli e di questo particolare cerchio; per tracciare la sua circonferenza basta usare quanto descritto nel paragrafo precedente scegliendo tre punti a piacere; nel disegno della figura 15 si sono scelti i punti medi dei lati.

Il codice usato per la figura 15 è il seguente, dove l'unica macro nuova è `\MiddlePointOf`.

```
\unitlength=0.01\linewidth
\begin{picture}(100,100)
\AutoGrid
\CopyVect20,0to\Pu \Pbox(\Pu)[t]{P_1}[2pt]
\CopyVect10,80to\Pd \Pbox(\Pd)[br]{P_2}[2pt]
\CopyVect100,60to\Pt \Pbox(\Pt)[l]{P_3}[2pt]
{\polygon(\Pu)(\Pd)(\Pt)\ignorespaces}
\TriangleMedianBase\Pu on\Pd and\Pt to\Pmu
\Pbox(\Pmu)[bl]{M_1}[2pt]
\TriangleMedianBase\Pd on\Pt and\Pu to\Pmd
\Pbox(\Pmd)[tl]{M_2}[2pt]
\TriangleMedianBase\Pt on\Pu and\Pd to\Pmt
\Pbox(\Pmt)[tr]{M_3}[2pt]
\ThreePointCircle(\Pmu)(\Pmd)(\Pmt)
\TriangleHeightBase\Pu on\Pd and\Pt to\Phu
\Pbox(\Phu)[b]{H_1}[2pt]
\TriangleHeightBase\Pd on\Pt and\Pu to\Phd
\Pbox(\Phd)[tl]{H_2}[2pt]
```

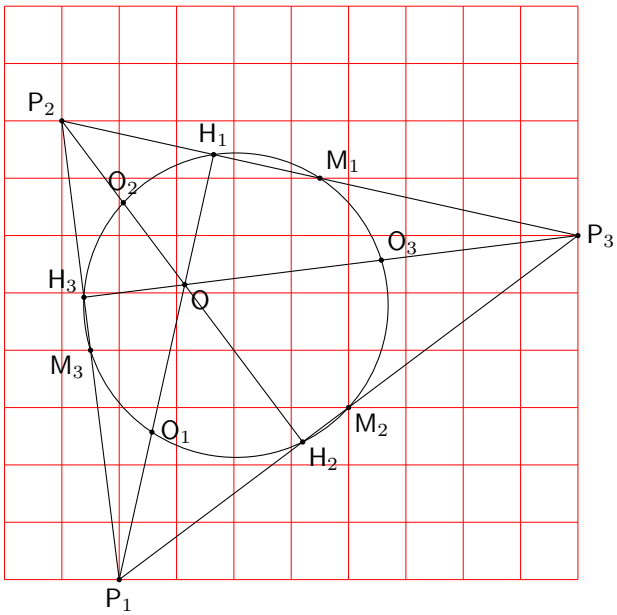

FIGURA 15: Cerchio dei nove punti

```
\TriangleHeightBase\Pt on\Pu and\Pd to\Pht
\Pbox(\Pht)[br]{H_3}[2pt]
\segment(\Pu)(\Phu)
\segment(\Pd)(\Phd)
\segment(\Pt)(\Pht)
\SubVect\Pu from\Phu to\DirHu
\SubVect\Pd from\Phd to\DirHd
\SubVect\Pt from\Pht to\DirHt
\IntersectionOfLines(\Pu)(\DirHu)
  and(\Pt)(\DirHt)to\Po
\Pbox(\Po)[tl]{O}[2pt]
\MiddlePointOf(\Po)(\Pu)to\Pou
\Pbox(\Pou)[l]{O_1}[2pt]
\MiddlePointOf(\Po)(\Pd)to\Pod
\Pbox(\Pod)[b]{O_2}[2pt]
\MiddlePointOf(\Po)(\Pt)to\Pot
\Pbox(\Pot)[bl]{O_3}[2pt]
\ThreePointCircle*(\Pou)(\Pod)(\Pot)
\end{picture}
```

E la macro nuova è definita così:

```
\def\MiddlePointOf(#1)(#2)to#3{%
\SubVect#1from#2to\Base
\ScaleVect\Base by0.5to\Base
\AddVect\Base and#1to#3\ignorespaces}
```

3.3.6 L'inellisse di Steiner

In un triangolo possono essere iscritte infinite ellissi tangenti internamente a tutti e tre i suoi lati. L'inellisse di Steiner, però, è unica; essa è definita come l'ellisse tangente internamente ai lati di un triangolo nei punti medi dei lati.

Esistono diverse dimostrazioni e costruzioni geometriche per costruirla, ma qui ho seguito una via percorribile con alcune macro che si possono aggiungere a quelle descritte in questo articolo.

L'enumerazione che segue mostra i punti su cui si basa il ragionamento.

1. Sia dato un triangolo T comunque orientato nel piano; si lavori su un triangolo equivalente T_0 ottenuto da T mediante rotazione, in modo che un lato sia orizzontale e il disegno del resto del triangolo si svolga nel semipiano superiore; è banale eseguire una traslazione e una rotazione di T per portarlo nella posizione specificata per T_0. Bisogna memorizzare l'ammontare della traslazione e della rotazione per riportare il tutto nella posizione iniziale alla fine della costruzione. In realtà, la traslazione e la rotazione non sarebbero necessarie per la costruzione; qui si usano per rendere più chiara la procedura per determinare questa ellisse.
2. Si costruisce il triangolo equilatero T_1 avente la stessa base di T_0; è evidente che l'inellisse di Steiner di questo triangolo equilatero è un cerchio e che questo è anche l'incerchio di centro C.
3. Si deformi T_1 fino a farlo diventare il triangolo isoscele T_2 di altezza pari all'altezza di T_0; si tratta di una affinità di deformazione verticale eseguita con un fattore di scala delle sole ordinate pari al rapporto fra l'altezza di T_0 e quella di T_1. Ne consegue che l'incerchio di T_1 si deforma nell'inellisse di cui l'asse verticale viene scalato con lo stesso fattore di scala delle ordinate, mentre l'asse orizzontale rimane pari al diametro dell'incerchio di $T1$. Per effetto di questa trasformazione il centro C si sposta nel punto C_e. Gli elementi per disegnare questa ellisse sono quindi tutti noti e la sua costruzione è immediata.
4. Bisogna ora deformare il triangolo isoscele T_2 con un movimento di *shear* o di scorrimento orizzontale proporzionale alle ordinate, in modo che si trasformi nel triangolo T_0; questa è una affinità di taglio che permette di muovere ogni punto del triangolo T_2 nel triangolo T_0; nel far questo, l'inellisse di T_2, di centro C_e, si trasforma nell'inellisse di T_0, di centro C_i, solo che gli assi della prima ellisse, che erano perpendicolari, si trasformano in due diametri obliqui dell'ellisse trasformata. Questo è quanto succede con l'affinità di taglio.
5. Si noti che l'affinità di taglio permette di calcolare direttamente le coordinate del centro della nuova ellisse, la cui ordinata rimane costante, ma l'ascissa si sposta di $b \tan \alpha$. Quindi si possono esprimere le equazioni delle ellissi rispetto a un sistema di coordinate con l'origine nel loro centro. La macro `\Xellisse` accetta sia le coordinate del centro, sia l'angolo di taglio, oltre ai semiassi, quindi non è un problema disegnare l'ellisse trasformata con i mezzi a disposizione nell'ambiente picture. Tuttavia, le equazioni da risolvere per determinare l'angolo di rotazione e i semiassi dell'inellisse sono piuttosto complesse; invece non è un problema quello di creare una macro come `\XSellisse`,

che vedremo fra poco, che attiva a propria volta una nuova macro di base `\Sellisse` per disegnare ellissi tenendo conto anche dello scorrimento. `\Sellisse` e `\XSellisse` rimangono retrocompatibili rispettivamente con `\ellisse` e `\Xellisse` e la `S` nel loro nome ricorda che gestiscono anche lo scorrimento.

6. L'unica informazione per gestire lo scorrimento consiste appunto nel determinare l'angolo α di cui si muovono le rette verticali per prendere la stessa inclinazione della mediana di T_0 rispetto alla sua base. La tangente di questo angolo rappresenta il parametro che gestisce l'affinità di scorrimento governata dalle due formule seguenti:

$$\begin{cases} x' = x + y\tan\alpha \\ y' = y \end{cases} \tag{1}$$

7. Conviene definire una macro, `\EllisseSteiner`, che riceva solo i vertici del triangolo e poi faccia da sola tutti i calcoli e le necessarie costruzioni. Conviene che disponga di una variante (selezionabile con un asterisco facoltativo) che permetta di costruire tutti passaggi intermedi, oppure che esponga solo il risultato finale.

8. L'ultimo passo è quello di ripristinare la posizione della costruzione, riportando il triangolo T_0 con la sua inellisse nella posizione originale.

La macro per eseguire i calcoli e disegnare il risultato finale, eventualmente con i passi intermedi della costruzione, si chiama `\EllisseSteiner` che verrà descritta fra poco; ma grazie a essa si può disegnare la costruzione dell'ellisse di Steiner come nella figura 16 con il codice

```
\unitlength=0.009\linewidth
\begin{picture}(100,100)
\AutoGrid
\EllisseSteiner*(10,10)(90,20)(70,80)[3]
\end{picture}
```

Senza ingombrare il disegno con la costruzione, si può disegnare il risultato finale nella figura 17 con il codice

```
\unitlength=0.009\linewidth
\begin{picture}(100,100)
\AutoGrid
\EllisseSteiner(10,10)(90,20)(70,80)[2]
\end{picture}
```

Come si vede, la differenza fra i due disegni è ottenuta solamente con un asterisco facoltativo per la macro `\EllisseSteiner`.

Il codice di `\EllisseSteiner` può sembrare complicato; ma altro non è se non quanto descritto sopra; si può seguire passo passo semplicemente rileggendo i punti dell'enumerazione precedente. Certo, è stato necessario aggiungere diverse istruzioni per fare o non fare certe cose a seconda della presenza dell'asterisco; tuttavia non confondono le idee circa il processo vero e proprio per determinare l'inellisse.

La macro ha la sintassi seguente:

`\EllisseSteiner`⟨`*`⟩(⟨*primo vertice*⟩)
(⟨*secondo vertice*⟩)
(⟨*terzo vertice*⟩)[⟨*diametro*⟩]

dove l'asterisco facoltativo serve per disegnare o non disegnare i passi della costruzione; seguono i tre vertici del triangolo presi nell'ordine tale che la traslazione e la rotazione iniziali portino la costruzione sull'asse delle ascisse e il resto del triangolo si svolga nel semipiano superiore. Infine, il ⟨*diametro*⟩ indica il diametro dei pallini neri con cui si marcano i punti salienti; se si omettono le dimensioni, si assume che siano indicati in punti tipografici. Questa impostazione facoltativa è interessante perché il pallino deve essere visibile indipendentemente dalla scala del disegno.

```
\NewDocumentCommand\EllisseSteiner%
  {s d() d() d() O{1}}{\bgroup
%
\IfBooleanTF{#1}{}{\put(#2)}{%
  \CopyVect0,0to\Pu
  \SubVect#2from#3to\Pd
  \SubVect#2from#4to\Pt
  \IfBooleanTF{#1}{}{%
    \Pbox(\Pu)[r]{P_1}[#5]\Pbox(\Pd)[t]{P_2}[#5]
    \Pbox(\Pt)[b]{P_3}[#5]}
  \ModAndAngleOfVect\Pd to\M and\Rot
  \MultVect\Pd by-\Rot:1 to\Pd
  \MultVect\Pt by-\Rot:1 to\Pt
  \IfBooleanTF{#1}{}{\rotatebox{\Rot}}%
  {\makebox(0,0)[bl]{%
    \IfBooleanTF{#1}{%
    \Pbox(\Pu)[r]{P_1}[#5]\Pbox(\Pd)[t]{P_2}[#5]
    \Pbox(\Pt)[b]{P_3}[#5]}{}%
  \polygon(\Pu)(\Pd)(\Pt)%
  \edef\B{\fpeval{\M/2}}%
  \edef\H{\fpeval{\B*tand(60)}}
  \IfBooleanTF{#1}{\Pbox(\B,\H)[b]{H}[#5]
  \polygon(\Pu)(\B,\H)(\Pd)}{}%
  \edef\R{\fpeval{\B*tand(30)}}
  \IfBooleanTF{#1}{\Pbox(\B,\R)[bl]{C}[#5]
    \Circlewithcenter\B,\R radius{\R}}{}%
  \GetCoord(\Pt)\Xt\Yt
  \edef\VScale{\fpeval{\Yt/\H}}
  \IfBooleanTF{#1}{%
    \polyline(\Pu)(\B,\Yt)(\Pd)
    \Pbox(\B,\Yt)[b]{V}[#5]}{}%
  \edef\Ce{\fpeval{\R*\VScale}}
  \IfBooleanTF{#1}{%
    \Xellisse(\B,\Ce){\R}{\Ce}
    \Pbox(\B,\Ce)[r]{C_e}[#5]
    \Pbox(\B,0)[t]{B}[#5]}{}%
  \SubVect\B,0 from\Pt to\SlMedian
  \IfBooleanTF{#1}%
    {\Dotline(\B,0)(\Pt){2}[1.5]}{}%
  \ModAndAngleOfVect\SlMedian to\Med and\Alfa
  \edef\Alfa{\fpeval{90-\Alfa}}
    \IfBooleanTF{#1}{%
      \Dotline(\B,\Yt)(\B,0){2}[1.5]
      \Pbox(\fpeval{\B+\Ce*tand{\Alfa}},%
      \Ce)[l]{C_i}[#5]
```

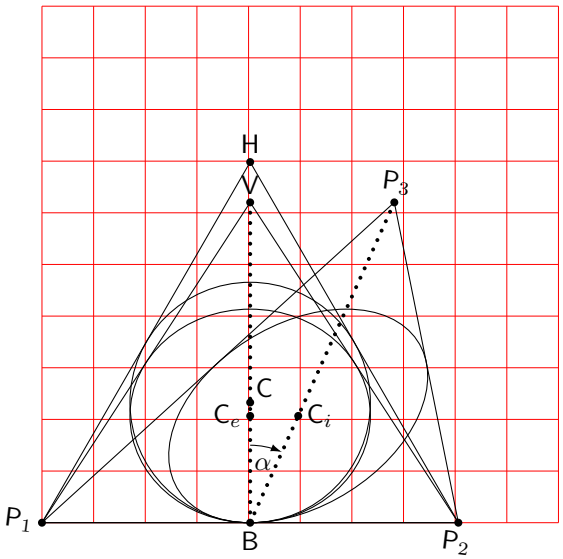

FIGURA 16: Sequenza delle operazioni per la determinazione dell'inellisse

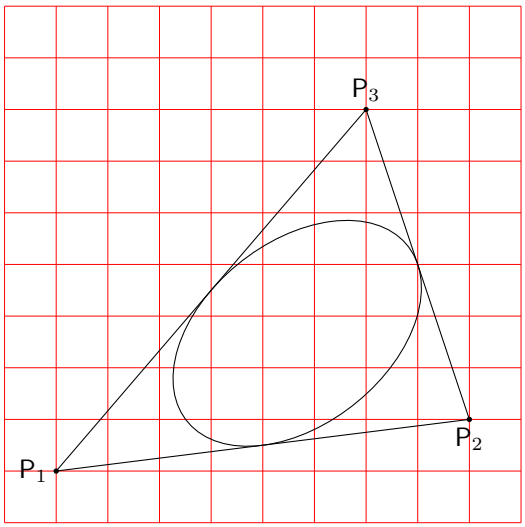

FIGURA 17: Inellisse finale

```
      \VectorArc(\B,0)(\B,15){-\Alfa}
      \Pbox(\fpeval{\B+2.5},14)[t]{\alpha[0]}{}%
  \edef\a{\R}\edef\b{\Ce}%
\CopyVect\fpeval{\B+\Ce*tand{\Alfa}},\Ce to\CI
\XSellisse(\CI)<\Alfa>{\R}{\Ce}
}}}\egroup\ignorespaces}
```

Tuttavia bisogna ancora disporre dei codici di `\Sellisse` e di `\XSellisse`.

Il codice di `\Sellisse` sembra terribilmente complicato rispetto al semplice codice di `\ellisse`; in realtà i comandi usati hanno nomi tali che si capisce subito a che cosa servono. Infatti, `\ScaleVector` è molto simile al comando `\ScaleVect` facente parte di curve2e; in effetti, il comando originale scala un numero complesso, quindi un vettore che va dall'origine a un punto dato; questo nuovo comando, invece, scala un vettore fra due punti dati. Il comando `\ShearVect`, invece, esegue l'affinità di scorrimento per un vettore fra due punti dati. I punti su cui operare sono gli stessi sui quali opera la macro `\ellisse` e sono dodici in totale, tre per ogni lato del rettangolo circoscritto all'ellisse prima di applicare l'affinità. Quindi, fra scalamenti e scorrimenti ci sono molti comandi da eseguire.

```
\def\ShearVect(#1)(#2)by#3to#4{%
\SubVect#1from#2to\AUX
\GetCoord(\AUX)\Aux\Auy
\edef\Aux{\fpeval{\Aux + #3*\Auy}}%
\edef\Auy{\fpeval{\Auy}}%
\AddVect\Aux,\Auy and#1to#4\ignorespaces}

\def\ScaleVector(#1)(#2)by#3to#4{%
\SubVect#1from#2to\AUX
\ScaleVect\AUX by#3to\AUX
\AddVect\AUX and#1to#4\ignorespaces}

\NewDocumentCommand\Sellisse{s m m O{0}}{\bgroup
\CopyVect#2,#3to\Ptr \ScaleVect\Ptr by-1to\Pbl
\CopyVect#2,-#3to\Pbr \ScaleVect\Pbr by-1to\Ptl
\edef\Ys{\fpeval{tand{#4}}}%
\edef\K{\fpeval{4*(sqrt(2)-1)/3}}%
%
\ShearVect(0,0)(0,#3)by\Ys to\Pmt
\ShearVect(0,0)(0,-#3)by\Ys to\Pmb
\ShearVect(0,0)(#2,0)by\Ys to\Pmr
\ShearVect(0,0)(-#2,0)by\Ys to\Pml
%
\ShearVect(\Pmr)(\Ptr)by\Ys to\Ptr
\ShearVect(\Pml)(\Ptl)by\Ys to\Ptl
\ShearVect(\Pmr)(\Pbr)by\Ys to\Pbr
\ShearVect(\Pml)(\Pbl)by\Ys to\Pbl
%
\IfBooleanTF{#1}{%
  \Pbox(\Ptr)[bl]{P_{tr}}
  \Pbox(\Pbl)[tr]{P_{bl}}
  \Pbox(\Pbr)[tl]{P_{br}}
  \Pbox(\Ptl)[br]{P_{tl}}
  \polygon(\Pbr)(\Ptr)(\Ptl)(\Pbl)}{}%
%
\ScaleVector(\Pmr)(\Ptr)by\K to\Crt
\ScaleVector(\Pmr)(\Pbr)by\K to\Crb
\ScaleVector(\Pml)(\Ptl)by\K to\Clt
\ScaleVector(\Pml)(\Pbl)by\K to\Clb
\ScaleVector(\Pmt)(\Ptr)by\K to\Ctr
\ScaleVector(\Pmt)(\Ptl)by\K to\Ctl
\ScaleVector(\Pmb)(\Pbr)by\K to\Cbr
\ScaleVector(\Pmb)(\Pbl)by\K to\Cbl
%
\IfBooleanTF{#1}{%
 \Pbox(\Crt)[l]{C_{rt}}\Pbox(\Crb)[l]{C_{rb}}
 \Pbox(\Clt)[r]{C_{lt}}\Pbox(\Clb)[r]{C_{lb}}
 \Pbox(\Ctr)[b]{C_{tr}}\Pbox(\Ctl)[b]{C_{tl}}
 \Pbox(\Cbr)[t]{C_{br}}\Pbox(\Cbl)[t]{C_{bl}}
%
\Pbox(\Pmr)[l]{P_{mr}}\Pbox(\Pmt)[b]{P_{mt}}%
\Pbox(\Pml)[r]{P_{ml}}\Zbox(\Pmb)[t]{P_{mb}}%
%
\polygon(\Pbr)(\Ptr)(\Ptl)(\Pbl)\thicklines}{}%
%
\moveto(\Pmr)
\curveto(\Crt)(\Ctr)(\Pmt)
\curveto(\Ctl)(\Clt)(\Pml)
\curveto(\Clb)(\Cbl)(\Pmb)
\curveto(\Cbr)(\Crb)(\Pmr)
\fillstroke
\egroup}
```

La sintassi di `\Sellisse` è la seguente:

```
\Sellisse⟨⋆⟩{⟨semiasse-a⟩}{⟨semiasse-b⟩}%
    <⟨shear⟩>
```

L'asterisco facoltativo permette di disegnare la costruzione, mentre senza di esso si ha solo il risultato finale; come la corrispondente macro `\ellisse`, questa nuova macro non dovrebbe essere usata direttamente, ma solo attraverso `\XSellisse`, che le trasferisce i dati essenziali. In alternativa, se l'utente prima si usarla specifica l'equivalenza di `\fillstroke` con `\strokepath`, l'ellisse che si ottiene ha il centro nell'origine degli assi. Nelle figure 18 e 19 si vedono appunto i due risultati.

Il codice di `\XSellisse` è leggermente più complesso di quello di `\Xellisse`, ma solo perché deve passare più argomenti alla macro subalterna `\Sellisse`.

```
\NewDocumentCommand\XSellisse%
  { s D(){0,0} O{0} D<>{0} m m s O{} o}%
  {\IfBooleanTF{#1}%
     {\let\fillstroke\fillpath}%
     {\let\fillstroke\strokepath}%
  \put(#2){\rotatebox{#3}{#8\relax
  \IfBooleanTF{#7}%
     {\Sellisse*{#5}{#6}[#4]}%
     {\Sellisse{#5}{#6}[#4]}%
  \IfValueTF{#9}%
    {\let\fillstroke\strokepath
    #9\Sellisse{#5}{#7}[#4]}{}}}%
  \ignorespaces}
```

La sua sintassi è la seguente:

```
\XSellisse⟨⋆⟩(⟨centro⟩)[⟨angolo⟩]%
    <⟨shear⟩>{⟨semiasse-a⟩}{⟨semiasse-b⟩}%
    ⟨⋆⟩[⟨impostazioni-1⟩]%
    [⟨impostazioni-2⟩]
```

Gli argomenti hanno gli stessi significati che hanno per la macro `\Xellisse`, tranne ⟨*shear*⟩ e il secondo asterisco facoltativo. Si noti che il parametro ⟨*shear*⟩, facoltativo e preimpostato a zero, indica l'angolo in gradi sessagesimali di cui ruotano le righe verticali soggette all'affinità di scorrimento; nella figura 18 si vede che è stato specificato il valore di 20°, e l'inclinazione dei lati obliqui del parallelogramma rispetto alla verticale è appunto di 20° positivi in senso orario. Il secondo asterisco, come si vede osservando la figura 18, che è stata composta con questo secondo asterisco, contiene tutta la costruzione alle spalle dell'affinità. Invece la figura 19, costruita senza il secondo asterisco, mostra solo il risultato finale.

3.3.7 Ellisse tangente internamente a un triangolo e di cui sia specificato un fuoco

Questo problema di geometria piana è ripreso dall'articolo di Estevão Candia Candia (2019), o meglio dalla sua tesi (Candia, 2018). Egli cita lo scritto di Sergio Alves (Alves, 2018) fornendo il riferimento che rimanda alla Rivista dei Professori di Matematica Brasiliani. Non sono riuscito ad accedervi, perché bisogna essere soci dell'associazione che la pubblica, ma ho trovato in rete un altro documento dello stesso autore e con lo stesso titolo *Elipses inscritas num triângulo* (vedi la nota nel riferimento Alves (2018)). In questo testo è presentato per gli allievi delle scuole secondarie superiori un certo numero di esercizi da svolgere con riga, squadra e compasso, fra i quali non c'è quello che riguarda questo paragrafo, ma ve ne sono le premesse. Nell'articolo di Candia (2019) c'è la costruzione completa della soluzione ottenuta con METAPOST. Qui mi interessa mostrare come la soluzione si possa trovare anche con i mezzi dell'ambiente picture esteso.

Il problema è il seguente: costruire l'ellisse tangente internamente ai lati di un triangolo dato, di cui sono noti i tre vertici e, per l'ellisse, un fuoco interno al triangolo. Ora è evidente che, se si riescono a determinare l'altro fuoco e i punti di tangenza, il problema è quasi risolto. Infatti, uno qualunque dei punti di tangenza permette di determinare la lunghezza dell'asse maggiore grazie alla definizione stessa dell'ellisse: essa è il luogo dei punti la cui somma delle distanze dai due fuochi è pari all'asse maggiore; se $2c$ è la distanza fra i due fuochi e $2a$ è la lunghezza dell'asse maggiore, allora è noto che l'asse minore $2b$ è legato agli altri due parametri dalla relazione:

$$a^2 = b^2 + c^2 \tag{2}$$

A questo proposito, quell'equazione si ricava a vista dal disegno della figura 20, dove il triangolo costruito intersecando la semicirconferenza superiore di raggio a con il semiasse maggiore interseca due punti che permettono di determinare il triangolo rettangolo POF_2 i cui cateti valgono b e c e l'ipotenusa vale a; Pitagora ci fornisce dunque la relazione (2) fra i semiassi e la semidistanza focale.

Determinare il centro dell'ellisse e l'inclinazione dell'asse maggiore conoscendo la posizione dei due fuochi è banale, per cui la macro `\Xellisse` può svolgere il proprio compito e disegnare l'ellisse cercata.

Merita osservare che, essendo noti i vertici del triangolo P_1, P_2 e P_3, su cui dovranno giacere i punti di tangenza, il primo fuoco F può essere collocato in una posizione qualsiasi all'interno di quel triangolo. Per questo motivo, ogni triangolo può contenere infinite ellissi tangenti internamente ai suoi lati. Ma, specificato un fuoco, l'ellisse cercata è unica.

Secondo le indicazioni indirettamente contenute in Alves (2018), ma esplicitamente indicate in Candia (2018), la determinazione del secondo fuoco e dei punti di tangenza avviene seguendo i seguenti passi.

```
\unitlength=0.009\linewidth
\begin{picture}(100,100)(-50,-50)
\AutoGrid
\Pbox(0,0)[rt]{O}[3]
\VECTOR(0,-50)(0,50)\Pbox(0,50){r}{y}[0]
\VECTOR(-50,0)(50,0)\Pbox(50,0){r}{x}[0]
\XSellisse<20>{40}{30}*
\end{picture}
\end{figure}
```

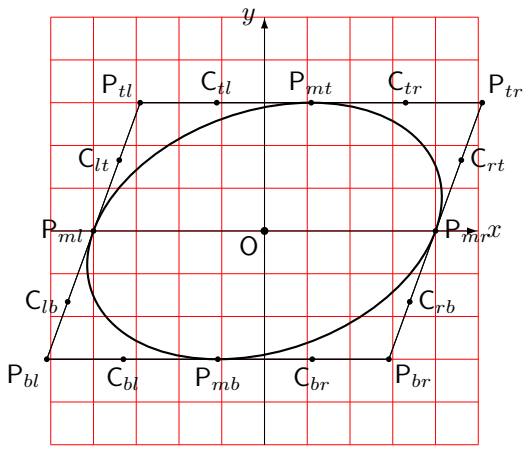

Figura 18: Costruzione di un'ellisse ottenuta con una affinità di scorrimento

```
\unitlength=0.009\linewidth
\begin{picture}(100,100)(-50,-50)
\AutoGrid
\Pbox(0,0)[rt]{O}[3]
\VECTOR(0,-50)(0,50)\Pbox(0,50){r}{y}[0]
\VECTOR(-50,0)(50,0)\Pbox(50,0){r}{x}[0]
\XSellisse<20>{40}{30}
\end{picture}
\end{figure}
```

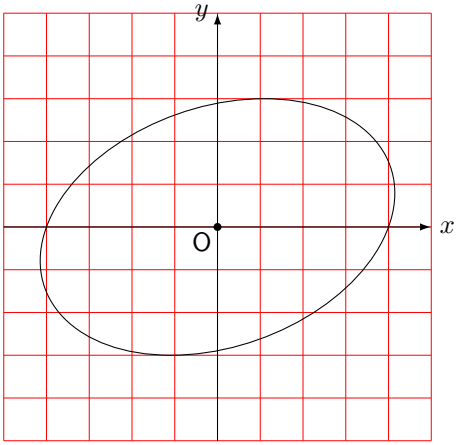

Figura 19: Una ellisse ottenuta con una affinità di scorrimento

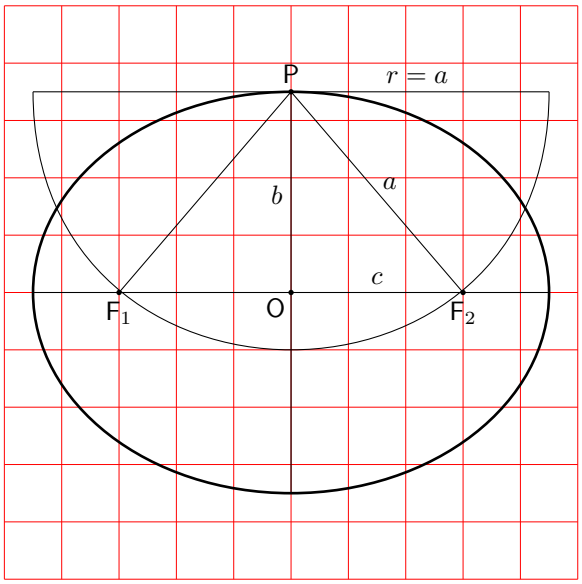

Figura 20: Legame fra gli assi e i fuochi

1. Si determinano i tre punti G_1, G_2 e G_3 esterni al triangolo, in modo che ciascuno sia il simmetrico di F rispetto a ciascun lato. Bisogna quindi determinare le perpendicolari ai lati che si dipartono da F e determinarne le intersezioni M_i con i lati stessi; questi rappresentano i punti medi fra F e ciascun punto G_i:

$$M_i = \frac{F + G_i}{2} \qquad \forall\, i = 1, 2, 3 \qquad (3)$$

Detto in altre parole, ciascun punto M_i è il punto di mezzo fra il fuoco e il corrispondente punto esterno al triangolo. Fra le macro già descritte, abbiamo già `\SegmentCenter`, ma non è questa quella che ci serve, perché qui l'incognita è un estremo del segmento; abbiamo bisogno della soluzione dell'equazione (3)

$$G_i = 2M_i - F \qquad \forall\, i = 1, 2, 3 \qquad (4)$$

Per ciascun valore di i si può usare invece una nuova macro `\SymmetricalPointOf` definita come

```
\def\SymmetricalPointOf#1respect#2to#3{%
  \ScaleVect#2by2to\Segm
  \SubVect#1from\Segm to#3\ignorespaces}
```

dove, nel nostro caso, il primo argomento è costituito dal fuoco, il secondo argomento dal punto medio, il terzo è la macro che riceve le coordinate del punto simmetrico.

2. Disponendo dei tre punti G_i simmetrici rispetto al fuoco, con la macro `\ThreePointCircle`

si può di disegnare un cerchio il cui centro costituisce il secondo fuoco F' dell'ellisse che si sta cercando. Si potrebbe evitare di disegnare questo cerchio esterno che la macro asteriscata `\ThreePointCircle` restituisce in `\C`. Quindi bisogna conservare questo valore in una macro il cui nome, possibilmente, ricordi che si tratta di F' (per esempio `\Fp`). Sarebbe ancora più conveniente usare la macro `\ThreePointCircleCenter` assegnando direttamente il centro calcolato alla macro `\Fp`. Ma qui vorremo anche disegnare la circonferenza, quindi è meglio usare la prima macro, che ci dà entrambi i risultati. Questa circonferenza appena descritta viene chiamata in alcune lingue *circonferenza direttrice*; si può constare che questa circonferenza ha molte proprietà interessanti; ma qui non è il caso di scendere nei particolari; basti sapere che essa ha il raggio pari all'asse maggiore $2a$ dell'ellisse. Volendo, quindi, si potrebbero saltare un paio dei punti seguenti in questa enumerazione.

3. I punti di tangenza T_1, T_2 e T_3, invece, sono le intersezioni con i tre lati dei tre raggi che vanno dal centro F' della circonferenza ai tre vertici G_i con i quali la si è costruita. E questa è un'altra proprietà della circonferenze direttrice.
4. Ora si dispone di tutti gli elementi per disegnare l'ellisse; il comando

   ```
   \SegmentCenter(\F)(\Fp)to\C
   ```

   permette di determinare il centro dell'ellisse; il comando

   ```
   \SegmentArg(\F)(\Fp)to\AngFocalAxis
   ```

   permette di determinare l'inclinazione dell'asse focale. Infine il comando

   ```
   \SegmentLength(\F)(\Fp)to\D
   ```

   dà la distanza focale, pari al doppio di c.
5. La macro `\SegmentLength` usata due volte permette di determinare le due distanze di uno qualsiasi dei punti di tangenza dai due fuochi; la loro semisomma dà a.
6. Noti a e c, la relazione (2) diventa la base per definire `\AxisFromAxisAndFocus` con la sintassi:

   `\AxisFromAxisAndFocus`⟨*argomento1*⟩ `and`⟨*argomento2*⟩ `to{`⟨*argomento3*⟩`}`

   e la sua definizione è la seguente:

   ```
   \def\AxisFromAxisAndFocus#1and#2to#3%
      {\fptest{abs(#1)>abs(#2)}%
      {\edef#3{\fpeval{sqrt(#1**2-#2**2)}}}%
      {\edef#3{\fpeval{sqrt(#2**2+#1**2)}}}}
   ```

   La macro è congegnata in modo tale che se il primo argomento è maggiore del secondo fornisce la differenza pitagorica, altrimenti fornisce la somma pitagorica; nel nostro caso a è il primo argomento ed è maggiore del secondo argomento c, quindi il risultato è la differenza pitagorica b.
7. Ora si hanno il centro `\C`, l'inclinazione dell'asse focale `\AngFocalAxis`, e i due semiassi `\a` e `\b`; quindi la macro `\Xellisse` può disegnare l'ellisse cercata.
8. Come nel caso dell'inellisse di Steiner, conviene creare un'unica macro che accetti un asterisco per disegnare tutta la costruzione geometrica o il solo risultato finale.

La figura 21 mostra appunto l'ellisse finale avendo usato il codice:

```
\unitlength0.0095\linewidth
\begin{picture}(150,150)(-30,-20)
\AutoGrid
\EllisseConFuoco(10,20)(120,-10)(0,100)(20,40)
\end{picture}
```

mentre la figura 22 mostra la costruzione per ottenere il risultato; il suo codice è il seguente:

```
\unitlength0.0095\linewidth
\begin{picture}(150,150)(-30,-20)
\AutoGrid
\EllisseConFuoco*(10,20)(120,-10)(0,100)(20,40)
\end{picture}
```

Come si vede, la macro `\EllisseConFuoco` con l'asterisco produce tutta la costruzione; togliendo l'asterisco, il triangolo e l'ellisse vengono disegnati senza le linee e i cerchi usati nella costruzione.

Non resta che esporre la definizione della macro `\EllisseConFuoco`, la cui sintassi è la seguente:

`\EllisseConFuoco`⟨⋆⟩`(`⟨*primo vertice*⟩`)%`
`(`⟨*secondo vertice*⟩`)(`⟨*terzo vertice*⟩`)(`⟨*fuoco*⟩`)`

dove l'asterisco facoltativo specifica se si vuole disegnare l'intera costruzione o se basta disegnare il risultato finale. Seguono le coordinate dei vertici del triangolo e le coordinate del fuoco, tutte racchiuse fra parentesi tonde. La macro che ora viene descritta contiene anche numerosi comandi `\Pbox` per etichettare i vari punti del disegno; i loro parametri facoltativi sono impostati per i disegni della figura 21 e, specialmente, della figura 22; potrebbero non essere ottimali per disegnare una costruzione diversa, cioè con un triangolo diverso e/o una diversa posizione del fuoco. Con un triangolo diverso e/o con il fuoco in un'altra posizione, la non ottimalità si potrebbe manifestare con le etichette dei punti sovrapposte ad alcune linee della costruzione; i pallini neri sono collocati correttamente, ma l'etichetta potrebbe richiedere allineamenti diversi. Spesso per vedere bene tutti gli elementi della costruzione basta ingrandirla a piena giustezza, invece che alla giustezza di una colonna. Tuttavia esistono anche altre vie.

Infatti, l'utente che voglia servirsi di questa macro potrebbe eliminare quasi tutti questi comandi `\Pbox` seguendo l'una o l'altra delle seguenti alternative.

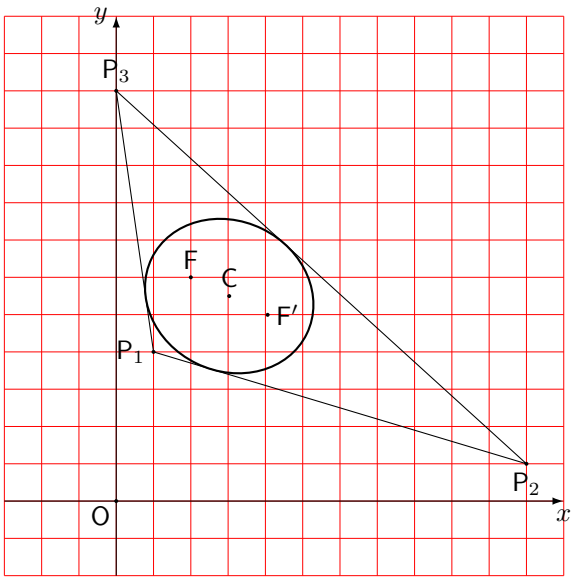

FIGURA 21: Un'ellisse tangente internamente a un triangolo, noto un fuoco dell'ellisse

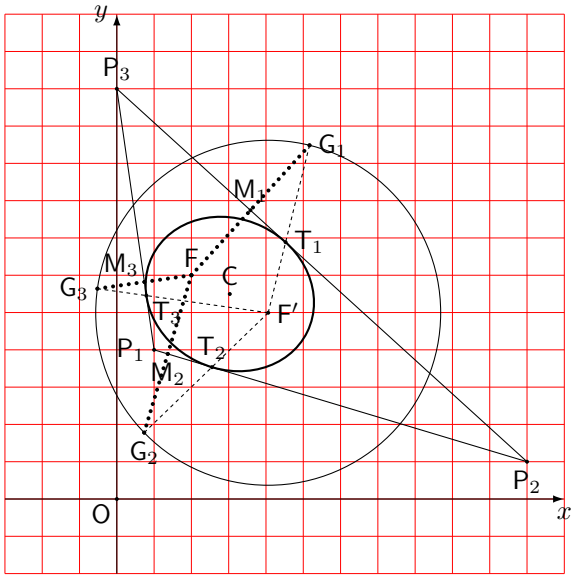

FIGURA 22: Costruzione dell'ellisse tangente internamente a un triangolo, noto un fuoco dell'ellisse

1. Etichetta a mano i punti che ritiene siano da marcare nel suo disegno specifico; la presenza della griglia agevola questo compito perché consente di stimare la posizione dei punti, sia pure in modo approssimato, per cui procedere a mano richiede diverse compilazioni in modo da aggiustare con precisione le coordinate dei punti da marcare.
2. Lascia i comandi `\Pbox` nella macro senza mettere nulla nel campo con l'etichetta, in modo che i pallini siano posti con precisione, poi mette a mano le etichette con altrettanti comandi `\Pbox` impostando a zero il diametro del pallino (ma specificando a mano che vuole comporre i simbolo del punto con il font senza grazie, per esempio `\Pbox(0,110)[b]{\mathsf P_3}[0]`), poiché, quando il diametro del pallino è nullo, l'etichetta viene composta in corsivo matematico. Aggiustare a mano la posizione dell'etichetta è più facile perché non richiede altrettanta precisione di quella richiesta per collocare il pallino al punto giusto.

```
\NewDocumentCommand\EllisseConFuoco%
  {s d() d() d() d()}{\bgroup%
\CopyVect#2to\Puini \CopyVect#3to\Pd
\CopyVect#2to\Pu
\CopyVect#3to\Pd
\CopyVect#4to\Pt
\CopyVect#5to\F
\polygon(\Pu)(\Pd)(\Pt)
\Pbox(\Pu)[r]{P_1}[1.5]
\Pbox(\Pd)[t]{P_2}[1.5]
\Pbox(\Pt)[b]{P_3}[1.5]
\Pbox(\F)[b]{F}[1.5]
\SegmentArg(\Pu)(\Pt)to\At
\SegmentArg(\Pu)(\Pd)to\Ad
\SegmentArg(\Pd)(\Pt)to\Au
\IntersectionOfLines(\Pu)(\At:1)
  and(\F)(\fpeval{\At+90}:1)to\Mt
\IntersectionOfLines(\Pd)(\Ad:1)
  and(\F)(\fpeval{\Ad+90}:1)to\Md
\IntersectionOfLines(\Pd)(\Au:1)
  and(\F)(\fpeval{\Au+90}:1)to\Mu
\IfBooleanTF{#1}%
 {\Pbox(\Mt)[br]{M_3}[1.5]
  \Pbox(\Md)[t]{M_2}[1.5]
  \Pbox(\Mu)[b]{M_1}[1.5]}{}
\SymmetricalPointOf\F respect\Mu to\Gu
\IfBooleanTF{#1}{\Pbox(\Gu)[l]{G_1}[1.5]}{}
\SymmetricalPointOf\F respect \Md to\Gd
\IfBooleanTF{#1}{\Pbox(\Gd)[t]{G_2}[1.5]}{}
\SymmetricalPointOf\F respect \Mt to\Gt
\IfBooleanTF{#1}{\Pbox(\Gt)[r]{G_3}[1.5]}{}
\IfBooleanTF{#1}%
  {\ThreePointCircle*(\Gu)(\Gd)(\Gt)}%
  {\ThreePointCircle(\Gu)(\Gd)(\Gt)}
\CopyVect\C to\Fp \Pbox(\Fp)[l]{F'}[1.5]
\IfBooleanTF{#1}%
  {\Dotline(\F)(\Gt){2}[1.5]
   \Dotline(\F)(\Gd){2}[1.5]
   \Dotline(\F)(\Gu){2}}[1.5]{}
\IntersectionOfSegments(\Pu)(\Pt)
  and(\Fp)(\Gt)to\Tt
\IntersectionOfSegments(\Pu)(\Pd)
  and(\Fp)(\Gd)to\Td
\IntersectionOfSegments(\Pd)(\Pt)
  and(\Fp)(\Gu)to\Tu
\IfBooleanTF{#1}%
  {\Pbox(\Tu)[l]{T_1}[1.5]
   \Pbox(\Td)[b]{T_2}[1.5]
   \Pbox(\Tt)[tl]{T_3}[1.5]
   \Dashline(\Fp)(\Gu){1}
   \Dashline(\Fp)(\Gd){1}
   \Dashline(\Fp)(\Gt){1}}{}
\DistanceAndDirOfVect\Fp minus\Tt
  to\DFp and\AFu
\DistanceAndDirOfVect\F  minus\Tt
  to\DF and\AF
\SegmentCenter(\F)(\Fp)to\CE
\Pbox(\CE)[b]{C}[1.5]
```

```
\edef\a{\fpeval{(\DFp+\DF)/2}}
\SegmentArg(\F)(\Fp)to\AngFocalAxis
\SegmentLength(\F)(\CE)to\c
\AxisFromAxisAndFocus\a and\c to\b
\Xellisse(\CE)[\AngFocalAxis]{\a}{\b}%
  [\thicklines]
\VECTOR(-30,0)(120,0)
\Pbox(120,0)[t]{x}[0]
\VECTOR(0,-20)(0,130)
\Pbox(0,130)[r]{y}[0]
\Pbox(0,0)[tr]{0}[1.5]
\egroup\ignorespaces}
```

4 Commenti

Questo articolo propone alcune tecniche che si possono usare per disegnare costruzioni geometriche che richiedano solamente l'uso di riga, squadra e compasso. Certamente con METAPOST alcune di queste costruzioni potrebbero essere più semplici, perché METAPOST è nato per lavorare con numeri complessi, ed è in grado di risolvere equazioni. In realtà è in grado di determinare le intersezioni non solo di segmenti, ma anche di curve di Bézier cubiche; siccome è nato da METAFONT, la precisione dei calcoli era sufficiente per disegnare i caratteri, ma non era, forse, sufficientemente precisa in altri casi.

Knuth, nel suo testo (Knuth, 1986) dice che in realtà lui usava impropriamente METAFONT dalla finestra comandi come se fosse una calcolatrice, tanto era soddisfatto dei calcoli che eseguiva; ma erano calcoli basati sull'aritmetica intera su cui sono fondati sia METAFONT, sia TEX. Con questo tipo di aritmetica, i numeri fratti sono trattati come numeri a virgola fissa, con 16 cifre binarie nella parte fratta; poi sono scalati mediante il fattore 2^{16} in modo che siano espressi come numeri interi; questa notazione a virgola fissa è usata in entrambi i programmi per esprimere le lunghezze al loro interno mediante il loro valore intero espresso in *scaled points*.

Inoltre la sintassi del linguaggio di METAPOST è decisamente diversa da quella di TEX e LATEX, per cui l'utente deve imparare un altro linguaggio di programmazione. Se questo valga la pena rispetto a quanto ho descritto in questo articolo è più una questione di gusti personali, che di efficienza e praticità d'uso. Tuttavia, personalmente, preferisco definire e usare macro in linguaggio TEX, tanto più che ora, con il linguaggio LATEX 3, le prestazioni sono quasi illimitate. Questo "quasi" dipende dal fatto che, per quanto io sappia, questo linguaggio ancora non consente di risolvere direttamente le equazioni. Anche i semplici sistemi lineari richiedono di determinare analiticamente le formule risolutive che poi sono facili da implementare con le funzionalità del pacchetto xfp. La precisione dei calcoli è certamente maggiore di quanto serva per la grafica.

Tuttavia, sono convinto che sia i docenti delle scuole secondarie di secondo livello, sia i docenti universitari dei corsi propedeutici di matematica e geometria possono trovare molto giovamento se, invece di dettare appunti (o lasciare che gli allievi prendano appunti), distribuissero in rete dei fascicoli, o dispense, delle loro lezioni; invece dei terribili formati EPUB, buoni per un romanzo, ma decisamente scadenti per un testo scientifico, è meglio che usino il formato standard PDF, in modo che gli allievi di oggi, informatizzati fin dai primi mesi di vita, possano sfruttare i loro potenti mezzi per disporre di scritti, composti, e disegnati come si deve.[10]

5 Ringraziamenti

Ringrazio di cuore Enrico Gregorio che mi ha aiutato molto a capire quello che si può fare con la libreria per il calcolo decimale in virgola mobile del linguaggio LATEX 3. Mi ha anche scritto un documento (incompiuto, ma per me sufficiente) che forse in futuro sostituirà la scarna documentazione di xfp oggi distribuita con il sistema TEX.

Ringrazio anche Francesco Biccari che, avendo desiderio di comporre carte millimetrate lineari e logaritmiche, mi ha stimolato ad approfondire la mia conoscenza della libreria di calcolo del linguaggio LATEX 3.

Riferimenti bibliografici

Alves, Sergio (2018). «Elipses inscritas num triângulo». *Revista do Professor de Matemática*, **96**. Questo è il sito da dove è possibile scaricare l'articolo solo da parte dei soci. L'articolo in formato PDF è però scaricabile dalla rete eseguendo una ricerca con la stringa "Sergio Alves Elipses inscritas num triangulo".

Beccari, Claudio (2018a). «Introduzione a xparse». *ArsTEXnica*, (26).

— (2018b). *PM-ISOmath – The poor-man ISO math bundle*. GUIT. Version 1.0.04. Leggibile con `texdoc pm-isomath`.

— (2020a). *The extension package curve2e*. GUIT. Version 2.0.8. Leggibile con `texdoc curve2e`.

— (2020b). «The curve2e manual». GUIT. Leggibile col comando `texdoc curve2e-manual`.

— (2020c). «The euclideangeometry package». PDF document. Leggibile col comando `texdoc euclideangeometry`.

— (2020d). «The euclideangeometry package manual». PDF document. Leggibile col comando `texdoc euclideangeometry-man`.

10. È bene ricordare che il formato PDF è quello prescritto dalle norme ISO per l'archiviabilità dei documenti in formato elettronico, (C.V. Radhakrishnan *et al.*, 2018); per comporre la matematica con le notazioni per la fisica e della tecnologia conformi con le norme ISO, vedano (Beccari, 2018b).

Candia, Estevão Vinicius (2018). *Matemática e o METAPOST*. Tesi di Dottorato, Universidade Federal de Mato Grosso do Sul.

— (2019). «A Brazilian Portuguese work on MetaPost, and how mathematics is embedded in it». *TUGboat*, **40** (3), pp. 247–250.

C.V. Radhakrishnan, Hàn Thế Thành, Ross Moore e Peter Selinger (2018). *Generation of PDF/X- and PDF/A- compliant PDFs with* `PdfTeX`. River Valley Technologies, Trivandrum, India. Versione 1.6. Leggibile con `texdoc pdfx`.

De Marco, Agostino (2009). «Produrre grafica vettoriale di alta qualità programmando `asymptote`». *ArsTEXnica*, (8), pp. 25–39.

— (2019). «Graphics for LaTeX users». *ArsTEXnica*, (28).

Fuster, Robert (2012). «The xpicture package – Several extensions of the picture standard environment – User Manual». PDF document. Leggibile col comando `texdoc xpicture`.

Gässlein, Hubert, Rolf NiePraschk e Josef Tkadlec (2016). *The pict2e package*. TUG. Version 0.3b. Readable with `texdoc pict2e`.

Hammerlind, Andy, John Bowman e Tom Prince (2018). *Asymptote: the vector graphics language*. TUG. Version 244. Readable with `texdoc asymptote`.

Hobby, John D. (2018). *METAPOST– A users' manual*. TUG. Version 2.00 (2.0rc2). Readable with `texdoc metapost`.

Knuth, Donald Ervin (1986). *The METAFONT book*. Addison Wesley, Reading, Mass.

Lamport, Leslie (1994). *A document preparation system — LaTeX — User's guide and reference manual*. Addison Wesley, Reading, Mass., 2ª edizione.

Matthes, Alain (2020). «AlterMundus — `tkz-euclide`». PDF document. Leggibile col comando `texdoc tkzdoc-euclide`. Il pacchetto è interessantissimo e ricopre in parte quanto descritto in questo articolo. Ma la documentazione, pur estesa, lascia a desiderare, con diversi esempi non documentati affatto; oltretutto con cambi continui di lingua dall'inglese al francese e viceversa.

Miklavec, Mojca (2013). *Using* `context` *and tikz terminals for gnuplot in ConTEXt*. Readable with `texdoc gnuplot`. This document gives instructions to install the external program `gnuplot`, version 4.6.0 or later, and to configure it to be used by the typesetting programs of the TEX system.

Tantau, Till (2019). *TikZ & PGF*. TUG. Version 3.1.1 – Readable with `texdoc tikz` or with `texdoc pgf` or with `texdoc pgfmanual`.

Van Zandt, Timothy (2003). *PSTricks – User's guide*. TUG. Version 97. Readable with `texdoc pstricks`.

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

Carta da regalo con LaTeX. Una proposta di gadget di benvenuto per il GUIT

Gianluca Pignalberi

Sommario

Diverse soluzioni programmative per carta da regalo similcommerciale con LaTeX.

Abstract

Several programming solutions for off-the-shelf-like gift paper with LaTeX.

Introduzione

In occasione del GUIT*meeting2019* abbiamo festeggiato gli 80 anni di Claudio Beccari.[1] Il Consiglio Direttivo, in rappresentanza del GUIT, gli ha fatto un piccolo dono.

Che regalo si può fare a un importante ex professore del Politecnico di Torino che ha speso oltre 30 anni al "servizio di TeX" (gli appassionati di fumetti avranno forse sentito qualcosa di familiare: Plazzi e Rosati (1996))? Il comitato ristretto[2] aveva proposto al Direttivo di regalare a Claudio un disegno di Duane Bibby: proposta accettata. Il disegno avrebbe dovuto ritrarre Claudio sotto la Mole Antonelliana attorniato da alcuni membri storici del GUIT intenti a festeggiarlo. Bibby, contattato da Maurizio Himmelmann (uno dei fondatori del GUIT), ha rifiutato il lavoro essendo ormai in pensione.

In quei giorni veniva lanciata su Kickstarter una raccolta fondi per rimettere in sesto una Linotype ubicata a Vienna (Kaps, 2018). Tra le ricompense per i *backer* c'era un set di biglietti da visita con testo di lunghezza massima prefissata e scelto da ogni sottoscrittore. Tale ricompensa sarebbe stata corredata delle tre righe di caratteri (Helvetica) di piombo fuse dalla Linotype ripristinata e usate per la stampa dei biglietti. Il nome dei sottoscrittori sarebbe stato poi "tatuato" sulla macchina, quindi il GUIT avrebbe avuto il suo riconoscimento tangibile per il contributo dato al progetto.

Figura 1: Il regalo di Claudio Beccari.

Ricevuto per tempo[3] il regalo (visibile nella figura 1), rimaneva il problema di presentarlo: la scatola era un'anonima scatola di cartone pressato grigio, realizzata alla bell'e meglio, coi punti metallici a tenere insieme gli angoli della scatola e del coperchio.

Tale dono, consegnato da Enrico Gregorio e gradito dal festeggiato (Beccari, 2019), andava incartato in una maniera… originale. Ci sarebbe voluta della bella carta da regalo. Ma perché comprarla, quando si poteva realizzarne di personalizzata con LaTeX? Avrebbe dovuto ricordare le carte commerciali con i disegni disposti a scacchiera, ma avere il logo del GUIT al posto di anonimi disegnini. Questa sì!, che sarebbe stata una carta originale.

All'insaputa di tutti decisi di verificare l'esistenza di un servizio di *print-on-demand* (POD d'ora in poi) che realizzasse tale prodotto. Trovatolo, stesi un documento LaTeX per comporre il foglio in maniera conforme alle specifiche richieste dal POD stesso. Le caratteristiche autoimposte per la carta erano le seguenti: sfondo colorato uniformemente; pattern composto dal logo GUIT disposto a scacchiera, variamente dimensionato (da `\large` a `\Huge`) e ruotato di multipli di 45°. Naturalmente, dimensioni e rotazioni sarebbero state casuali.

L'articolo descrive il problema affrontato, cioè la realizzazione della carta da regalo personalizzata GUIT, partendo dall'analisi delle specifiche richieste dal servizio POD (sezione 1). Quindi mostra e commenta le prove preliminari in LaTeX (sezione 2) e il successivo completamento in C delle parti

1. In realtà, il suo ottantesimo compleanno cade nel 2020, ma abbiamo preferito approfittare del suo ultimo meeting da direttore di ArsTeXnica nel "suo" Politecnico.

2. Quella dei comitati ristretti è un'idea di Luigi Scarso: tre o quattro membri del Direttivo delegati per uno specifico compito, magari supportati da un volontario non facente parte del Direttivo, tengono riunioni informali tra loro anche senza preavviso per mettere a punto la strategia per portare a termine il compito. La cosa ha funzionato per il meeting e per la domanda all'Anvur.

3. I tempi previsti per il ripristino della Linotype, la realizzazione e la consegna delle ricompense erano pochi mesi: entro gennaio 2019; il regalo è arrivato però alla fine di luglio 2019 a causa di alcuni intoppi legati proprio al ripristino della Linotype.

casuali (sezione 3). Passa poi a descrivere il passaggio finale per l'invio in stampa (sezione 4). Di seguito fornisce una traduzione in LuaLATEX (sezione 5), una in LATEX (sezione 6) e una in LATEX3 (sezione 7) del codice C + LATEX. Dopodiché generalizza il progetto originale della sezione 3, prima adottando un'immagine al posto del logo testuale (sezione 8), poi (sezione 9) rendendo parametriche le dimensioni della tabella (numero di righe e colonne) per adeguare la griglia alle esigenze dell'utente e a quelle della pagina, col conseguente ricalcolo delle dimensioni delle celle. Nelle conclusioni (sezione 10) lancia al nuovo Direttivo una proposta. L'appendice A, destinata solo ai programmatori e agli interessati, parla del sostanziale fallimento dei tentativi di migliorare pur di poco le prestazioni del codice C proposto alla sezione 3. Il suo contenuto prevede solo un minimo di conoscenze sull'aritmetica dei puntatori nel C. Infine, l'appendice B esegue un confronto orientativo dei programmi descritti nelle sezioni 3–7.

1 Il servizio POD

Prima di iniziare questo progetto, è stata fondamentale la ricerca di un servizio POD in grado di realizzare il prodotto voluto. Una rapida ricerca sul web ha fornito il nome di *print24* (PRINT24, 2019), servizio in grado di produrre da uno a n fogli di carta da regalo personalizzati, in diversi pesi e formati.

Nel modulo d'ordine è possibile stabilire il formato del foglio (A0–A3, B1 e B2), il suo orientamento, il peso della carta (115, 135 o 170 g/m^2) e la sua proprietà (opaca o lucida), il tipo di stampa (nero, nero più un Pantone, nero più oro o argento, CMYK o CMYK più uno degli speciali appena nominati) e altri dati necessarî a determinare il costo finale.

Nel caso in esame, la scelta è stata: foglio A3 a singola facciata, orientato orizzontalmente e carta da 115 g/m^2 opaca.

2 Il documento preliminare

Il sorgente LATEX per la produzione di un foglio con un pattern posizionato a griglia è in sé piuttosto banale: si tratta di riempire una tabella adeguatamente dimensionata alternando le celle vuote e piene in base alle righe e alle colonne. Sappiamo da Beccari *et al.* (2016) che il problema può essere risolto con minori o maggiori "compattezza" ed "espressività" computazionale in base al linguaggio scelto. Nel caso esaminato in questo articolo, la tabella sarà una vera tabella "di testo" e non un disegno e il pattern di riempimento sarà un logo di testo anziché una campitura di colore.

Il primo passo è stato fatto scrivendo una tabella 3×3, logo non ruotato e monodimensionale il cui risultato è visibile nella figura 2:

```
1 \documentclass[12pt]{standalone}
2
```

Figura 2: Risultato della compilazione della versione preliminare della carta da regalo.

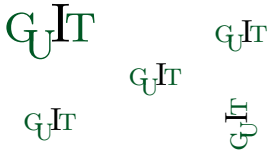

Figura 3: Risultato della compilazione della versione preliminare della carta da regalo con ridimensionamento e rotazione del logo.

```
3  \usepackage[T1]{fontenc}
4  \usepackage[utf8]{inputenc}
5  \usepackage{guit}
6
7  \begin{document}
8  \noindent\begin{tabular}{@{}ccc@{}}
9  \GuIT* & & \GuIT* \\
10 & \GuIT* & \\
11 \GuIT* & & \GuIT* \\
12 \end{tabular}
13 \end{document}
```

Prima di espandere la tabella alle dimensioni finali, la sperimentazione è andata avanti per verificare gli altri vincoli progettuali: la rotazione e il ridimensionamento del logo. Ecco il nuovo contenuto della tabella coi nuovi numeri di riga dopo l'inclusione di graphicx per avere il comando `\rotatebox`:

```
9  \LARGE\GuIT* & & \GuIT* \\
10 & \GuIT* & \\
11 \GuIT* & & \rotatebox[origin=c]{90}{\
       GuIT*} \\
```

Il nuovo risultato è visibile nella figura 3

Fin qui non c'è niente di particolarmente difficile: è anzi tutto banale. Riempire una tabella dalle dimensioni definitive di 20×16,[4] riempiendone le celle dispari in una riga e quelle pari nell'altra riga è solo un po' più lungo: ovviamente basta scrivere per esteso le prime due righe e copiarne il contenuto nelle altre. La vera sfida è fare in modo che ogni logo sia dimensionato e ruotato casualmente, sebbene con dimensioni e angoli prefissati: farlo a mano per una tabella di nove caselle è accettabile; per una tabella di 320 caselle lo è meno, volendo mantenere la casualità. Senza contare che lo stesso metodo potrebbe essere applicato alla produzione di carta da parati e il numero delle celle della ta-

4. Queste dimensioni sono state decise pensando a fogli di formato A4 e A3 e sono state mantenute fisse per tutto il progetto. Naturalmente, nessuno ne impone la fissità e vedremo come stabilirle arbitrariamente nella sezione 9.

bella potrebbe essere ancora maggiore rispetto a quante ne può contenere un già grande foglio A0.

Un'altra cosa di cui tener conto è la dimensione verticale delle celle, che non è possibile demandare all'intestazione della tabella.

3 La versione definitiva: C + LaTeX

Il linguaggio C e stato il mio strumento di lavoro dal quinto anno di scuola superiore fino a qualche anno dopo il termine della mia ormai lontana carriera di studente universitario. Nonostante la ruggine, il ricorso al C è stata la scelta più logica per risolvere la parte mancante in breve tempo (neanche due settimane tra il momento della nascita dell'idea della carta e la data di consegna del regalo).

Il progetto completo consta di due file preliminari (un generatore di dati in C, `makeGPtv1.c`, e un utilizzatore di dati in LaTeX, `giftpaper.tex`), e del file di dati (`table.tex`) atto a "collegare" i due preesistenti. Verrà dunque applicata la tecnica descritta esaustivamente in Giacomelli e Pignalberi (2015).

Il primo file preliminare, che è anche l'unico a essere eseguito, contiene questo codice:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <time.h>
5
6  #define DEGSTEP 45
7  #define MAXROTSTEP 8
8  #define NUMDIM 5
9  #define NROWS 16
10 #define NCOLS 20
11 #define MAXLEN 255
12
13 FILE *open_file (char *, char *);
14
15 int main ()
16 {
17   short i, j, rotation, dimpos;
18   char *dimen[NUMDIM] = {"\\large", "
         \\Large", "\\LARGE", "\\huge", "
         \\Huge"},
19        tabular[NROWS][NCOLS][MAXLEN],
              fg[] = "table.tex", tmp[
              MAXLEN];
20   FILE *table;
21
22   srand (time (NULL));
23   for (i = 0; i < NROWS; i++)
24     for (j = 0; j < NCOLS - 1; j++)
25       strcpy (tabular[i][j], "␣&");
26   for (i = 0; i < NROWS; i++)
27     strcpy (tabular[i][NCOLS - 1], "␣
           \\\\");
28   for (i = 0; i < NROWS; i++)
29     for (j = i % 2; j < NCOLS; j += 2)
           {
30       rotation = ((int) rand() %
             MAXROTSTEP ) * DEGSTEP;
31       dimpos = (int) rand() % NUMDIM;
32       sprintf (tmp, "\\begin{minipage
             }{2.1cm}\\vbox␣to␣1.856cm{\\
             vfill\\hfil\\rotatebox[
             origin=c]{%d}{%s\\GuIT*}\\
             hfill\\vfill}\\end{minipage}
             ␣%s", rotation, dimen[dimpos
             ], tabular[i][j]);
33       strcpy (tabular[i][j], tmp);
34     }
35   if ((table = open_file (fg, "w")) ==
         NULL) exit (1);
36   for (i = 0; i < NROWS; i++)
37     for (j = 0; j < NCOLS; j++)
38       fprintf (table, "%s", tabular[i
             ][j]);
39   fclose (table);
40   system ("pdflatex␣giftpaper");
41   return 0;
42 }
43
44 FILE * open_file (char *nomefile, char
       *accesso)
45 {
46   FILE *fp;
47
48   if ((fp = fopen (nomefile, accesso))
         == NULL)
49     fprintf (stderr, "Errore␣nell'
           apertura␣di␣%s\n", nomefile);
50   return fp;
51 }
```

Il programma si apre col caricamento di alcune librerie standard e con la definizione di alcune costanti (righe 1–11). La funzione `open_file`, dichiarata all'inizio (riga 13) e definita alla fine (righe 44–51), serve solo ad aprire un file comunicandone l'esito. La funzione `main`, dopo aver dichiarato e/o definito alcune variabili (tra cui gli array[5] contenenti la matrice che replica in memoria l'intera tabella dei pattern e il vettore delle possibili dimensioni del logo), svolge il grosso del lavoro. Dapprima inizializza il generatore di numeri pseudocasuali (riga 22), quindi inizializza la matrice: i primi due cicli (righe 23–25 e 26–27) inseriscono rispettivamente i separatori di cella (`&`) nelle celle delle prime $n-1$ colonne e i separatori di riga (\\) nelle celle dell'ultima colonna. Il ciclo alle righe 28–34 riempie la matrice con i dati effettivi: il primo `for` scandisce tutte le righe dell'array

5. Non bisogna dimenticare che in C gli array iniziano con la cella nº 0. Normalmente vengono chiamati "vettori" se monodimensionali e "matrici" se pluridimensionali, ma è chiaro che un vettore di stringhe è, di fatto, una matrice di caratteri.

(cioè si muove in verticale dall'alto verso il basso); il secondo scandisce le colonne (cioè si muove orizzontalmente) saltando una posizione ogni due e partendo da 0 o da 1 in base al valore della riga (riga 29 del codice; l'Appendice approfondirà proprio questo ciclo); nella cella identificata a ogni passo del ciclo verrà scritto (riga 33) il codice LaTeX che dimensiona correttamente la cella e ci pone il logo con l'angolo di rotazione e la dimensione generati alle righe 30 e 31[6] completato col terminatore precedentemente scritto nella cella (riga 32). Le righe 35–39 scaricano il contenuto della matrice in memoria nel file `table.tex` e la riga 40 demanda al sistema operativo la compilazione del master `.tex`, il cui codice è riportato più avanti. Infine, la riga 41 termina l'esecuzione dando un codice 0, valore standard per segnalare un'esecuzione andata a buon fine.

Essendo questo un file C, andrà compilato seguendo le istruzioni del proprio compilatore. La sua esecuzione genera come sottoprodotto il file contenente la tabella da porre sul foglio e di cui mostriamo un esempio delle prime due celle delle prime due righe:

```
1 \begin{minipage}{2.1cm}\vbox to 1.856
      cm{\vfill\hfil\rotatebox[origin=c
      ]{0}{\Huge\GuIT*}\hfill\vfill}\end
      {minipage} & &
2 & \begin{minipage}{2.1cm}\vbox to
      1.856cm{\vfill\hfil\rotatebox[
      origin=c]{315}{\large\GuIT*}\hfill
      \vfill}\end{minipage} &
```

Quest'ultimo è il file "di raccordo" e viene incluso in `giftpaper.tex`, così da avere la tabella perfettamente composta. Come si può notare, le dimensioni orizzontali delle celle sono esattamente un ventesimo del lato maggiore di un foglio A3, mentre le dimensioni verticali sono un sedicesimo del lato minore dello stesso foglio. Le dimensioni sono precalcolate perché il progetto non è nato per essere modulare. Nella sezione 9 vedremo come rendere più versatile il prodotto.

Il contenuto di `giftpaper.tex` è il seguente:

```
1  \documentclass[12pt]{article}
2
3  \usepackage[T1]{fontenc}
4  \usepackage[utf8]{inputenc}
5  \usepackage{guit}
6  \usepackage{tabu}
7  \usepackage[a3paper,hmargin=0pt,
       vmargin=0pt,landscape]{geometry}
8  \usepackage[center,width=42.4cm,height
       =30.1cm]{crop}
9  \usepackage{xcolor}
10 \usepackage{graphicx}
11
12 \begin{document}
13 \pagecolor{black!5!yellow!30}
14 \noindent\begin{tabu} to \textwidth {@
       {}*{20}X}
15 \input{table}
16 \end{tabu}
17 \end{document}
```

A tabular, usato nella sezione precedente, è stato preferito tabu per la possibilità di specificare la larghezza della tabella e di equidimensionare le singole celle. Non sono stati usati gli specificatori di posizione orizzontale e verticale dell'ambiente tabu: per la presenza di minipage e `\vbox` nel codice da scrivere nelle celle sono stati usati direttamente comandi di TeX. Questo rende l'uso di tabu forse eccessivo e gli si potrebbe preferire tabularx.

Sicuramente si nota il fatto che la tabella inizia esattamente sul bordo della gabbia di pagina (cioè sul bordo sinistro del foglio; lo spessore naturale delle celle di una tabella si elimina scrivendo `@{}` sul lato della cella da cui lo si vuole eliminare) ma non termina sul bordo destro. Questo è voluto perché i logo alle dimensioni massime finiscono troppo vicino al margine destro, rovinando la simmetria della tabella rispetto al foglio.

Il risultato dell'esecuzione del programma in C è un PDF col colore di sfondo scelto e una serie di logo GUIT (a colori) dimensionati e ruotati casualmente posti a scacchiera su una griglia regolare posta su un foglio di dimensione totale A3 più le abbondanze richieste dal POD.

4 Postproduzione per la stampa

Dopo aver prodotto il PDF definitivo, che il POD vuole senza crocini di taglio o altri segni e visibile nella figura 4, rimane da applicare la proprietà relativa al modello di colore. La richiesta del POD è un file PDF, JPEG o TIFF a 300 dpi minimo e nella modalità colore CMYK con profilo colore Fogra39L per la stampa su carta patinata o Fogra47L per la stampa su carta riciclata o usomano.

Non avendo a disposizione un programma commerciale tipo InDesign, ho importato il PDF di LaTeX in un documento Scribus e ho esportato quest'ultimo come PDF dopo aver selezionato la modalità stampa (contrapposta alle modalità schermo/web e toni di grigio) e impostato l'unico profilo colore disponibile nell'installazione standard (Fogra27L, dato per uso con carta patinata). Il file è stato accettato senza alcun problema dal POD e il risultato è decisamente conforme alle aspettative o comunque abbastanza buono.

5 La traduzione in LuaLaTeX

Gli utenti di LuaTeX (LuaTeX development team, 2019) troveranno oggettivamente più sem-

6. Avremmo potuto risparmiare qualche byte di memoria e qualche millisecondo di tempo di accesso scrivendo il codice che genera i due valori direttamente nella `sprintf`, ma ciò a scapito della leggibilità del codice.

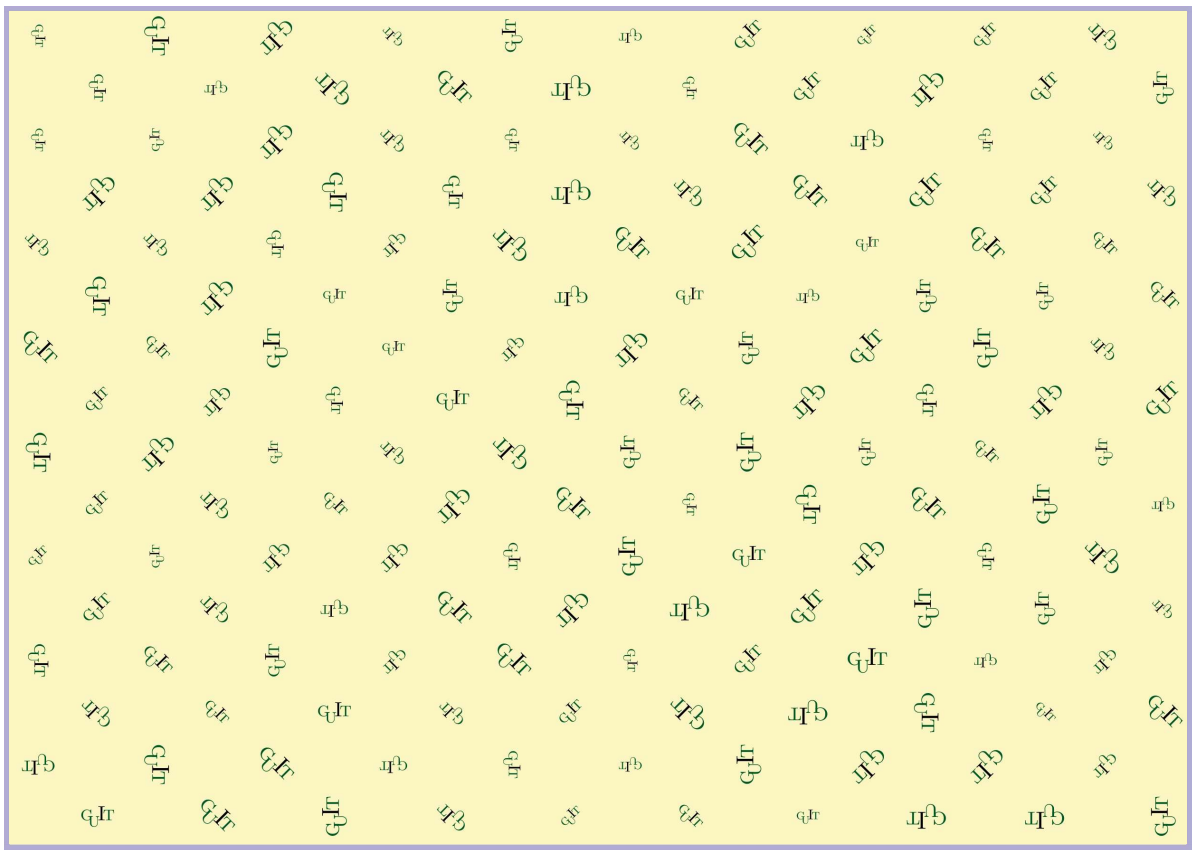

FIGURA 4: Il foglio definitivo mandato in stampa. Le abbondanze sono evidenziate artificialmente solo a beneficio dei lettori.

plice risolvere il problema grazie a Lua. Sebbene io abbia sentito parlare di Lua per la prima volta nel 2001 durante il mio periodo come borsista al CASPUR (ora parte del CINECA), non ho mai studiato né utilizzato tale linguaggio. Dunque, questa sezione descrive la mia prima esperienza d'uso di Lua puro e applicato.

Come già visto nella sezione 3, dobbiamo rendere pseudocasuale l'angolo di rotazione e la dimensione del logo di testo. Ci serviranno dunque due funzioni: una che generi un numero casuale $N \in \{0, 45, 90, 135, 180, 225, 270, 315\}$ e una che restituisca una dimensione compresa tra `\large` e `\Huge`. Come ampiamente discusso nell'implementazione originale in C, quest'ultima funzione si realizza generando un numero casuale che fungerà da indice dell'array in cui abbiamo memorizzato le stringhe coi comandi delle dimensioni volute. Il resto è banale applicazione delle chiamate a Lua. Il codice per ottenere la carta da regalo è il seguente:

```
\documentclass[12pt]{article}

\usepackage{fontspec}
\usepackage{guit}
\usepackage{tabu}
\usepackage[a3paper,hmargin=0pt,
    vmargin=0pt,landscape]{geometry}
\usepackage[center,width=42.4cm,height
    =30.1cm]{crop}
\usepackage{xcolor}
\usepackage{graphicx}
\usepackage{luacode}

\begin{luacode}
local dimen = {"\\large", "\\Large",
    "\\LARGE", "\\huge", "\\Huge"}
local Trows = 16
local Tcols = 20
local anglestep = 45
local mt = {}
function randrot()
  tex.print(math.random(0,7) *
      anglestep)
end
function randimen()
  tex.print(dimen[math.random(5)])
end
function maketable()
  for i = 0, Trows - 1 do
    mt[i] = {}
    for j = 0, Tcols - 2 do
      mt[i][j] = " &"
    end
  end
  for i = 0, Trows - 1 do
    mt[i][Tcols - 1] = " \\\\ "
  end
  for i = 0, Trows - 1 do
```

```
35      for j = i % 2, Tcols - 1, 2 do
36        mt[i][j] = "\\cell" .. mt[i][j]
37      end
38    end
39    tab = io.open ("table.tex", "w")
40    for i = 0, Trows - 1 do
41      for j = 0, Tcols - 1 do
42        tab:write (mt[i][j])
43      end
44    end
45    tab:close ()
46  end
47  \end{luacode}
48
49  \newcommand\cell{\begin{minipage}{2.1
        cm}\vbox to 1.856cm{\vfill\hfil\
        rotatebox[origin=c]{\directlua{
        randrot()}}{\directlua{randimen()}
         \GuIT*}\hfill\vfill}\end{minipage
        }}
50  \begin{document}
51  \pagecolor{black!5!yellow!30}
52  \directlua{maketable()}
53  \noindent\begin{tabu} to \textwidth {@
        {}*{20}X}
54  \input{table}
55  \end{tabu}
56  \end{document}
```

La riga 10 include il pacchetto per avere un ambiente che faciliti la scrittura del codice Lua; le righe 12 e 47 aprono e chiudono questo ambiente. Le righe 13–23 contengono il codice Lua discusso in precedenza: la funzione `randrot` fornisce un angolo di rotazione compreso tra 0° e 315° a intervalli di 45°; la funzione `randimen` restituisce una delle dimensioni comprese nella tabella[7] `dimen`. Infine, le righe 24–46 definiscono una funzione che, al pari di quanto visto nella sezione 3, inizializza una tabella Lua con `&` in tutte le colonne meno l'ultima (righe 25–30), dove scrive `\\` (righe 31–33), quindi antepone a questi caratteri il comando `\cell` solo dove occorre (celle pari delle righe pari e celle dispari delle righe dispari, righe 34–38); infine scrive l'array in un file esterno che verrà incluso entro l'ambiente tabu (righe 39–45).

Per evitare di scrivere un codice tanto lungo quanto poco manutenibile, definiamo il comando `\cell` (riga 49) nel modo affine a quello descritto nella sezione 3. Con LuaLATEX, però, possiamo fare le chiamate `\directlua` per generare i valori pseudocasuali nella macro di LATEX e dimenticare il metodo bizantino richiesto dal C, cioè generare una stringa col comando e i valori pseudocasuali da scrivere direttamente nella tabella, senza possibilità di condensare tutto in un comando LATEX. Quando all'interno della tabella dobbiamo riempire una cella, ci basterà dare il comando `\cell`.

7. Qui il termine tabella si riferisce al tipo di dato Lua e non a un ambiente `tabular` di LATEX.

6 Non conosci il pacchetto lcg?

Arrivati a questo punto, molti degli utenti principianti e senza esperienza di programmazione potrebbero obiettare che le soluzioni offerte finora sono troppo esoteriche e potrebbero volere una soluzione in LATEX puro. Un Egregio (o Esperto) Guru potrebbe allora indicare la via dicendo: «Perché vuoi uccidere le mosche col C(annone) o col raggio l(u)aser quando puoi usare un comodo schiacciamosche? Non conosci il pacchetto lcg?» e dimostrando così la sua conoscenza enciclopedica dell'ecosistema di TEX e la mia miopia non solo ottica (e poi mi rimanderebbe pure a Stack Exchange (2015) per farmi valutare una soluzione basata su Ti*k*Z).

Effettivamente, il pacchetto in questione contiene delle macro per generare numeri pseudocasuali che verranno memorizzati in un contatore definibile dall'utente. Anche a beneficio dei non programmatori C o Lua, diamo una soluzione al problema basata su lcg.

Il codice, che trovate in `giftpaperlatex.tex` nei sorgenti di accompagnamento, è il seguente:

```
1   \documentclass[12pt]{article}
2
3   \usepackage[T1]{fontenc}
4   \usepackage{guit}
5   \usepackage{tabu}
6   \usepackage[a3paper,hmargin=0pt,
        vmargin=0pt,landscape]{geometry}
7   \usepackage[center,width=42.4cm,height
        =30.1cm]{crop}
8   \usepackage{xcolor}
9   \usepackage{graphicx}
10  \usepackage{lcg}
11
12  \makeatletter
13  \def\nloop#1{%
14    \def\nl@@p##1##2\repeat#1{%
15    \def##1{##2\relax\expandafter##1\fi}
          %
16     ##1\let##1\relax}%
17  \expandafter\nl@@p\csname nl@@p-\
        string#1\endcsname
18  }
19  \makeatother
20  \newcommand\dimension[1]{%
21    \ifcase #1\relax
22    \or \large
23    \or \Large
24    \or \LARGE
25    \or \huge
26    \or \Huge
27    \fi}
28  \newcounter{x}
29  \newcommand\cell{\begin{minipage}{2.1
        cm}\vbox to 1.856cm{\vfill\hfil\
        chgrand [first=0, last=7]\setbox
        1=\hbox{\rand}\hskip-\wd1\multiply
        \value{rand} by 45 \value{x}=\
```

```
      value{rand}\rotatebox[origin=c]{\
      value {x}}{\chgrand[first=1, last
      =5]\setbox1=\hbox{\rand}\hskip-\wd
      1\dimension{\value{rand}}\GuIT*}\
      hfill\vfill} \end{minipage}}
30
31  \begin{document}
32  \newwrite\tempfile
33  \immediate\openout\tempfile=table.tex
34  \pagecolor{black!5!yellow!30}
35  \newcounter{Trows}\setcounter{Trows
        }{16}
36  \newcounter{Tcols}\setcounter{Tcols
        }{20}
37  \newcounter{nrows}
38  \newcounter{ncols}
39  \value{nrows}=0
40  \nloop\a
41    \value{ncols}=0
42    \nloop\b
43      \ifodd\value{nrows}%
44        \ifodd\value{ncols}\immediate\
              write\tempfile{\unexpanded{\
              cell}}\fi
45      \else%
46        \ifodd\value{ncols}\relax\else\
              immediate\write \tempfile{\
              unexpanded{\cell}}\fi
47      \fi
48      \stepcounter{ncols}
49      \ifnum\value{ncols}<\value {Tcols
            }\immediate\write \tempfile{ &}
            %
50    \repeat\b
51      \immediate\write\tempfile{\
            unexpanded{\\}}\stepcounter{
            nrows}
52    \ifnum\value{nrows}<\value{Trows}%
53  \repeat\a
54  \immediate\closeout\tempfile
55  \noindent\begin{tabu} to \textwidth {@
        {}*{20}X}
56  \input{table}
57  \end{tabu}
58  \end{document}
```

Al pari delle versioni precedenti, il documento mette insieme le celle della tabella in un file (`table.tex`, soluzione proposta da Reza Afzalan su https://tex.stackexchange.com/questions/50296/problem-with-using-loop-inside-the-tabular-environment) che verrà poi incluso nell'ambiente tabu per la composizione. Vediamo nel dettaglio come fa. Alla riga 10 include il pacchetto lcg. Alle righe 12–18 definisce una versione di `\loop` (proposta da David Carlisle su https://tex.stackexchange.com/questions/58049/nested-loop-macro) che permette gli annidamenti. Alle righe 20–27 fornisce una macro (`\dimension`) che, in base al valore numerico ricevuto in input, emette un comando di dimensionamento il cui "valore" è compreso tra `\large` e `\Huge` (simuliamo in questo modo quanto abbiamo finora fatto con gli array o con le tabelle). Alla riga 28 viene creato un contatore (`x`), usato nel successivo comando `\cell` (scritto alla riga 29). Quest'ultimo, al netto di quanto visto anche nelle occasioni precedenti (dimensionamento orizzontale e verticale della cella, centratura della scritta) fa le seguenti cose: imposta la generazione di un numero compreso tra 0 e 7 (comando `\chgrand`) e lo genera (comando `\rand`). Questo numero, moltiplicato per 45 e messo nel contatore `x`,[8] viene usato come angolo per la `\rotatebox`. Dopodiché viene impostata la generazione di un numero compreso tra 1 e 5 che sarà l'argomento del comando `\dimension`. Alle righe 32–33 si definisce un file temporaneo (`table.tex`) che viene aperto in scrittura. Le righe 35–36 definiscono due contatori contenenti il numero totale di righe e colonne della tabella, queste ultime scandite grazie ai due contatori creati alle righe 37–38. I cicli annidati lavorano in questo modo: si parte dalla prima riga scandendo tutte le sue celle; se la riga e la colonna sono entrambe pari (ciò si verifica nelle celle (0,0), (0,2), (0,4)…) o entrambe dispari (cioè (1,1), (1,3), (1,5)…), si scrive sul file un comando `\cell` e, se non abbiamo riempito l'ultima colonna, un divisore di colonna `&`; dopo l'ultima colonna si emette `\\` e si prosegue alla nuova riga; se la riga appena scandita è ultima, il ciclo termina e il file `table.tex` viene chiuso.

Qualcuno si chiederà perché il comando `\rand` viene messo in una scatola la cui larghezza è poi usata per spostarsi a sinistra nel testo. Come avrete visto (e avrete letto nel manuale di lcg) l'uso del numero generato è conseguente alla sua generazione. Sebbene nel manuale non sia esplicitato (ma si vede negli esempî), l'uso di `\rand` inserisce uno spazio non richiesto nel testo e ciò implicherebbe lo spostamento a destra delle scritte, prima centrate nelle celle della tabella.

Un'ultima osservazione può essere fatta sull'inefficienza di scrivere il contenuto della singola cella o del divisore nel file. Sarebbe molto più comodo e veloce comporre un'unica stringa col contenuto di una riga e poi scrivere quest'ultima all'interno del file. Inoltre, il modo scelto mette su una riga il contenuto di una singola cella. Questo si riflette in qualche modo sull'efficienza? Ne discuteremo brevemente nell'appendice B. I lettori potranno cimentarsi con la miglioria proposta partendo dal codice fornito con l'articolo e dal suggerimento dato in https://tex.stackexchange.com/questions/74707/how-to-concatenate-strings-into-a-single-command.

8. Se pensate di usare direttamente il contenuto di `rand` come angolo per la `\rotatebox`, vi ritroverete frustrati perché, quando `\value{rand}` viene effettivamente usato, non è stato ancora moltiplicato per 45 nella catena di espansione delle macro.

7 L'alternativa... nativa: LaTeX3

Il Guru della precedente sezione si sarà chiesto finora perché non ho pensato subito a risolvere il problema con LaTeX3. La risposta, banale, è la stessa di LuaTeX: non ne avevo esperienza e non volevo rischiare di sforare i tempi di produzione. Dunque anche questa sezione rappresenta la mia prima esperienza di programmazione in LaTeX3 e senz'altro la soluzione proposta potrà essere migliorata.

Il codice, contenuto nel file `giftpaper3.tex`, è una mera traduzione del precedente LaTeX ed è il seguente:

```
1  \documentclass[12pt]{article}
2
3  \usepackage[T1]{fontenc}
4  \usepackage{guit}
5  \usepackage{tabu}
6  \usepackage[a3paper,hmargin=0pt,
       vmargin=0pt,landscape]{geometry}
7  \usepackage[center,width=42.4cm,height
       =30.1cm]{crop}
8  \usepackage{xcolor}
9  \usepackage{graphicx}
10 \usepackage{expl3}
11
12 \ExplSyntaxOn
13 \cs_new:Npn \dimension #1
14 { \if_case:w #1 \relax
15     \or: \large
16     \or: \Large
17     \or: \LARGE
18     \or: \huge
19     \or: \Huge
20   \fi}
21 \cs_new:Npn \cell
22 {\begin{minipage}{2.1cm}\vbox to 1.856
       cm{\vfill\hfil\rotatebox[origin=c
       ]{\fp_eval:n {45*randint(0,7)}}{\
       dimension{\fp_eval:n{randint(1,5)
       }}\GuIT*}\hfill\vfill}\end{
       minipage}}
23 \ExplSyntaxOff
24
25 \begin{document}
26 \pagecolor{black!5!yellow!30}
27 \ExplSyntaxOn
28 \iow_new:N \tempfile
29 \iow_open:Nn \tempfile {table.tex}
30 \int_new:N\Trows\int_set:Nn\Trows{16}
31 \int_new:N\Tcols\int_set:Nn\Tcols{20}
32 \int_new:N\nrows\int_set:Nn \nrows{0}
33 \int_new:N\ncols
34 \int_do_while:nNnn{\nrows}<{\Trows}{%
35   \int_set:Nn\ncols{0}
36   \int_do_while:nNnn{\ncols}<{\Tcols}{
         %
37     \int_if_odd:nTF{\nrows}{%
38       \int_if_odd:nTF{\ncols}{%
39         \iow_now:Nn \tempfile {\cell}
40       }{\relax}
41     }{%
42         \int_if_odd:nTF\ncols{\relax}{
               %
43           \iow_now:Nn \tempfile {\cell}
44         }%
45     }
46     \int_incr:N\ncols
47     \int_compare:nNnTF{\ncols}<{\Tcols
           }{%
48       \iow_now:Nn\tempfile{ &}
49     }{\relax}
50   }
51   \iow_now:Nn \tempfile {\\}
52   \int_incr:N\nrows
53 }
54 \iow_close:N \tempfile
55 \ExplSyntaxOff
56 \noindent\begin{tabu} to \textwidth {@
       {}*{20}X}
57 \input{table}
58 \end{tabu}
59 \end{document}
```

La riga 10 include il pacchetto che abilita l'uso delle funzioni[9] di LaTeX3 e, dopo averne abilitato la sintassi (riga 12), definiamo le due funzioni `\dimension` (righe 13–20) e `\cell` (righe 21–22). Queste fanno esattamente quel che facevano le omonime funzioni nella versione LaTeX, ma si avvantaggiano delle funzioni di nuova introduzione. In particolare, `\dimension` trae vantaggio da `\if_case:w`, l'equivalente di `\ifcase`, mentre `\cell` usa la funzione `\fp_eval:Npn` e il generatore di numeri pseudocasuali entro un intervallo (`randint`) messi a disposizione dal modulo per i calcoli in virgola mobile di LaTeX3. Alla riga 23 disabilitiamo la sintassi di LaTeX3.

Al pari della versione LaTeX, anche qui ho preferito scrivere la sequenza di operazioni direttamente nel corpo del documento, senza ricorrere a una funzione come nel caso (obbligato) della versione LuaLaTeX. Detta sequenza, come di consueto aperta con `\ExplSyntaxOn` alla riga 27 e chiusa con `\ExplSyntaxOff` alla riga 55, è l'esatta traduzione di quanto visto e spiegato nella sezione 6 e che non ripeterò. Potete comunque fare riferimento a The LaTeX3 Project (2019) per ogni informazione sul significato dei comandi usati, sulla loro interfaccia e sugli esempî d'utilizzo.

8 Solo testo? E le immagini?

Esaurita la digressione sui linguaggi "altri", torniamo al mio caro vecchio C tenendo in mente che tutte le modifiche di questa sezione sono imple-

9. Ricordiamo essere questa la nomenclatura adottata da LaTeX3.

mentabili "gratis" o quasi nei sorgenti in LuaTEX, in LATEX e in LATEX3.

Possiamo pretendere di scrivere un progetto del genere, pur semplice e breve, e costringerlo a usare solo il logo del GUIT? Ovviamente no!, e questa sezione mostrerà come fare a usare un'immagine al posto del logo.

I cambiamenti principali da fare rispetto al sorgente originale sono due: 1) il vettore delle dimensioni non dovrà più contenere dei comandi LATEX ma dei fattori di scala, quindi dei valori *floating point* (per comodità saranno sempre cinque valori, compresi tra 0.75 e 1.25) e non delle stringhe di caratteri; 2) il comando che forma la stringa da mettere in ogni cella non conterrà più il comando del logo ma un `\includegraphics`:

```
18    float dimen[NUMDIM] = {.75, .875, 1,
          1.125, 1.25};

32        sprintf (tmp, "\\begin{minipage
              }{2.1cm}\\vbox␣to␣1.856cm{\\
              vfill\\hfil\\rotatebox[
              origin=c]{%d}{\\
              includegraphics[scale=%f]{
              tux.pdf}}\\hfill\\vfill}\\
              end{minipage}␣%s", rotation,
               dimen[dimpos], tabular[i][j
              ]);
```

L'immagine usata nella figura 5 è il Tux così come disponibile in Scribus.

Il file completo si trova nei sorgenti col nome di `makeGPiv1.c`.

9 Via, verso nuove dimensioni!

Finora abbiamo lavorato con fogli A3, la dimensione minima prevista dal POD di riferimento. È ora di rendere possibile la selezione delle altre dimensioni disponibili. Lasciare immutate le dimensioni della tabella comporterà produrre fogli più grandi troppo vuoti, ma anche troppo pieni se si deciderà di produrne di più piccoli. Dunque sarà opportuno, se non ingrandire o rimpicciolire le dimensioni del testo o del disegno da porre sul foglio, almeno dare la possibilità di variare arbitrariamente il numero di righe e di colonne della tabella. In questa sezione applicheremo questa modifica al file C, che sarà di conseguenza adattato anche per calcolare le dimensioni fisiche delle celle, laddove prima tali dimensioni erano prefissate. Per completezza potremo scegliere di includere del testo o un'immagine, integrando dunque le due tipologie di oggetti visti separatamente nelle sezioni 3 e 8.

Il sorgente, contenuto nel file `makeGP.c` e stavolta commentato direttamente solo nei punti di nuova introduzione, è il seguente:

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   #include <time.h>
5   #include <ctype.h>
6
7   #define DEGSTEP 45
8   #define MAXROTSTEP 8
9   #define NUMDIM 5
10  #define NROWS 16
11  #define NCOLS 20
12  #define MAXLEN 255
13
14  FILE *open_file (char *, char *);
15  int max (int, int);
16
17  int main (argc, argv)
18    int argc;
19    char *argv[];
20  {
21  /* argv[1]: flag testo/immagine.
        Valori leciti: T e I
22   * argv[2]: formato del foglio. Valori
         leciti: A0, A1, A2, A3, B1, B2
23   * argv[3]: numero di righe (celle
        verticali)
24   * argv[4]: numero di colonne (celle
        orizzontali)
25   * argv[5]: testo (anche comando) o
        nome immagine */
26  /* Controllo preliminare sul numero di
        argomenti: 5 parametri + il
       comando */
27    if (argc < 6) {
28      printf ("Parametri␣insufficienti.␣
          Uso:\n␣makeGP␣<tipo␣di␣pattern
          >␣(T␣o␣I,␣testo␣o␣immagine)␣<
          formato␣carta>␣(A0,␣A1,␣A2,␣A3,
          ␣B1,␣B2)␣<numero␣celle␣
          verticali>␣<numero␣celle␣
          orizzontali>␣<testo␣(se␣
          comando,␣\\␣dev'essere␣
          raddoppiato;␣es.␣\\\\textcolor
          {red}{A})␣o␣nome␣del␣file␣con␣
          l'immagine>");
29  /* Se gli argomenti inseriti sono
       insufficienti si termina con
       codice -1 */
30      exit (-1);
31    }
32  /* nRows e nCols prendono il numero di
        righe e colonne della tabella
       dalla riga di comando. Se negativi
        o nulli, tali valori sono portati
        a 1 */
33    short i, j, rotation, dimpos, nSheet
         = 9, nRows = max(atoi(argv[3])
        ,1), nCols = max(atoi(argv[4])
        ,1);
34    float cellWidth, cellHeight;
35  /* Solo la prima lettera del pattern
       è significativa */
```

FIGURA 5: Il foglio di carta da regalo con un'immagine al posto di un logo testuale.

```
36    char tabular[nRows][nCols][MAXLEN],
          fg[] = "table.tex", tmp[MAXLEN],
           patternName[MAXLEN/10], pattern
           = toupper(argv[1][0]),
          sheetType[2], realPattern[MAXLEN
          /2];
37    struct sheets {
38      char latexname[MAXLEN/10];
39      float sheetWidth, sheetHeight;
40    };
41  /* Questa variabile contiene le
       dimensioni dei fogli selezionabili
        */
42    struct sheets sheet[6] = {
43      {"a0paper", 118.9, 84.1}, {"
            a1paper", 84.1, 59.4},
44      {"a2paper", 59.4, 42}, {"a3paper",
             42, 29.7},
45      {"b1paper", 100, 70.7}, {"b2paper",
             70.7, 50}};
46    FILE *table;
47  /* Errore sul primo parametro. Uscita
       con codice 1 */
48    if ((pattern != 'T') && (pattern !=
          'I')) {
49      printf ("Tipo␣sconosciuto.␣Sono␣
            ammissibili␣solo␣T␣(testo)␣e␣I
            ␣(immagine)\n");
50      exit (1);
51    }
52    strcpy(sheetType, argv[2]);
53    strcpy(realPattern, argv[5]);
54    srand (time (NULL));
55  /* Recuperiamo le dimensioni del
       foglio per il successivo calcolo
       delle dimensioni reali delle celle
       . Se il formato è sconosciuto, il
       programma termina con codice 1 */
56    if (!strcmp (sheetType, "a0") || !
          strcmp (sheetType, "A0")) nSheet
          =0;
57    if (!strcmp (sheetType, "a1") || !
          strcmp (sheetType, "A1")) nSheet
          =1;
58    if (!strcmp (sheetType, "a2") || !
          strcmp (sheetType, "A2")) nSheet
          =2;
59    if (!strcmp (sheetType, "a3") || !
          strcmp (sheetType, "A3")) nSheet
          =3;
60    if (!strcmp (sheetType, "b1") || !
          strcmp (sheetType, "B1")) nSheet
          =4;
61    if (!strcmp (sheetType, "b2") || !
          strcmp (sheetType, "B2")) nSheet
          =5;
62    if (nSheet == 9) {
63      printf ("Formato␣di␣pagina␣
            sconosciuto.\n");
64      exit (1);
```

```
  }
  cellWidth = sheet[nSheet].sheetWidth / nCols;
  cellHeight = sheet[nSheet].sheetHeight / nRows;
  for (i = 0; i < nRows; i++)
    for (j = 0; j < nCols - 1; j++)
      strcpy (tabular[i][j], " &");
  for (i = 0; i < nRows; i++)
    strcpy (tabular[i][nCols - 1], " \\\\");
  for (i = 0; i < nRows; i++)
    for (j = i % 2; j < nCols; j += 2) {
      rotation = ((int) rand() % MAXROTSTEP ) * DEGSTEP;
      dimpos = (int) rand() % NUMDIM;
/* A seconda che il pattern di stampa sia testo o immagine, il vettore delle dimensioni conterrà dei comandi LaTeX o dei fattori di scala. Le dimensioni della minipage e della vbox sono quelle calcolate in precedenza a partire dalla dimensione del foglio e dal numero di righe e colonne */
      if (pattern == 'T') {
        char *dimen[NUMDIM] = {"\\large", "\\Large", "\\LARGE", "\\huge", "\\Huge"};
        sprintf (tmp, "\\begin{minipage}{%fcm}\\vbox to %fcm{\\vfill\\hfil\\rotatebox[origin=c]{%d}{%s %s}\\hfill\\vfill}\\end{minipage} %s", cellWidth, cellHeight, rotation, dimen[dimpos], realPattern, tabular[i][j]);
      }
      else {
        float dimen[NUMDIM] = {.75, .875, 1, 1.125, 1.25};
        sprintf (tmp, "\\begin{minipage}{%fcm}\\vbox to %fcm{\\vfill\\hfil\\rotatebox[origin=c]{%d}{\\includegraphics[scale=%f]{%s}}\\hfill\\vfill}\\end{minipage} %s", cellWidth, cellHeight, rotation, dimen[dimpos], realPattern, tabular[i][j]);
      }
      strcpy (tabular[i][j], tmp);
    }
  if ((table = open_file (fg, "w")) == NULL) exit (1);
  for (i = 0; i < nRows; i++)
    for (j = 0; j < nCols; j++)
      fprintf (table, "%s", tabular[i][j]);
  fclose (table);
/* Prepara il comando di shell per sostituire i dati relativi al foglio nel documento LaTeX */
  sprintf (tmp, "sed -i giftpaper.tex -e '7s/..paper/%s/g'", sheet[nSheet].latexname);
  system (tmp);
/* Prepara il comando di shell per sostituire i dati relativi alle abbondanze nel documento LaTeX */
  sprintf (tmp, "sed -i giftpaper.tex -e '8s/width=[0-9]*[.]*[0-9]*cm, height=[0-9]*[.]*[0-9]*cm/width=%.1fcm,height=%.1fcm/g'", sheet[nSheet].sheetWidth + .4, sheet[nSheet].sheetHeight + .4);
  system (tmp);
/* Prepara la stringa-comando per aggiornare il numero di colonne del file .tex */
  sprintf (tmp, "sed -i giftpaper.tex -e '14s/{[0-9][0-9]*}/{%d}/g'", nCols);
/* ... e la esegue */
  system (tmp);
  system ("pdflatex giftpaper");
  return 0;
}

FILE * open_file (char *nomefile, char *accesso)
{
  FILE *fp;

  if ((fp = fopen (nomefile, accesso)) == NULL)
    fprintf (stderr, "Errore nell' apertura di %s\n", nomefile);
  return fp;
}

/* Restituisce il valore massimo tra due interi */
int max (int x, int y)
{
  if (x > y) return x;
  return y;
}
```

Per ottenere il risultato della figura 4, ma col pattern ovviamente diverso per via della casualità, ci basta dare il comando

```
makeGP T A3 16 20 \\GuIT*
```

A questo punto ci fermiamo, ma la fantasia potrebbe scatenarsi ulteriormente: un vettore potrebbe contenere due o più nomi di immagine e uno

o più testi, da selezionare casualmente allo stesso modo delle dimensioni; si potrebbero programmare i flag in input in modo da non essere più costretti a ricordarci la sequenza esatta da dare alla riga di comando; anche un *help* non sarebbe male; pur poco significativa, la possibilità di avere il foglio con orientazione *portrait* invece della *landscape* scelta inizialmente potrebbe essere una miglioria, così come lo sarebbe eliminare i nomi dei file *legacy* e renderli parametrici, cioè far scegliere dalla riga di comando il nome del documento LaTeX da compilare e quello della tabella da includere. Ma lo scopo dell'articolo non è certo questo. Ci siamo già spinti molto oltre e lo faremo di più, ma in altra direzione, nell'appendice A.

Rimane da vedere come fare la stessa cosa per le altre versioni. Partiamo da alcune osservazioni preliminari. Tutte le versioni "statiche" mostrano che le misure dell'abbondanza sono espresse in termini assoluti, così come le misure delle celle. Sarebbe un'ottima cosa se crop permettesse di specificare un formato di carta e un'abbondanza; purtroppo non lo permette, preferendo far inserire delle misure assolute. In `makeGP.c` abbiamo notato come dobbiamo far eseguire un comando che cambi le misure dell'abbondanza (righe 96–97) in `giftpaper.tex` in accordo col foglio scelto. Questa cosa, accettabile laddove il cambiamento sia automatico, non è una cosa molto intelligente da fare nelle versioni alternative: potremmo non ricordare a memoria né le dimensioni del formato scelto né le dimensioni dell'abbondanza. Allora occorre trovare una strategia che ci permetta di non inserire tali dati all'atto dell'inclusione di crop nelle versioni alternative a quella in C + LaTeX. Quando cambiamo il numero di righe e colonne da porre sulla tabella, dobbiamo ricordarci di cambiarne anche l'intestazione. Farlo a mano può essere fonte di dimenticanza. Quindi vedremo come fare per cambiare meno cose possibili nelle versioni in LuaLaTeX, LaTeX e LaTeX3. Partiamo dalla versione LuaLaTeX.

```
\documentclass[12pt]{article}

\usepackage{fontspec}
\usepackage{guit}
\usepackage{tabu}
\usepackage[a3paper,hmargin=0pt,
    vmargin=0pt,landscape]{geometry}
\newdimen\cropwidth
\newdimen\cropheight
\cropwidth=\paperwidth
\cropheight=\paperheight
\advance\cropwidth by 4mm
\advance\cropheight by 4mm
\usepackage[center,width=\the\
    cropwidth,height=\the\cropheight]{
    crop}
\newcount\Trows
\Trows=16
\newcount\Tcols
\Tcols=20
\usepackage{xcolor}
\usepackage{graphicx}
\usepackage{luacode}

\begin{luacode}
local dimen = {"\\large", "\\Large",
    "\\LARGE", "\\huge", "\\Huge"}
local Trows = tex.count['Trows']
local Tcols = tex.count['Tcols']
local anglestep = 45
local pw = tex.dimen['paperwidth']
local ph = tex.dimen['paperheight']
local mt = {}
function randrot()
  tex.print(math.random(0,7) *
      anglestep)
end
function randimen()
  tex.print(dimen[math.random(5)])
end
function pwidth()
  tex.print(pw / Tcols .. "sp")
end
function pheight()
  tex.print(ph / Trows .. "sp")
end
function maketable()
  for i = 0, Trows - 1 do
    mt[i] = {}
    for j = 0, Tcols - 2 do
      mt[i][j] = " &"
    end
  end
  for i = 0, Trows - 1 do
    mt[i][Tcols - 1] = " \\\\"
  end
  for i = 0, Trows - 1 do
    for j = i % 2, Tcols - 1, 2 do
      mt[i][j] = "\\cell" .. mt[i][j]
    end
  end
  tab = io.open ("table.tex", "w")
  for i = 0, Trows - 1 do
    for j = 0, Tcols - 1 do
      tab:write (mt[i][j])
    end
  end
  tab:close ()
end
\end{luacode}

\newcommand\cell{\begin{minipage}{\
    directlua{pwidth()}}\vbox to \
    directlua{pheight()}{\vfill\hfil\
    rotatebox[origin=c]{\directlua{
    randrot()}}{\directlua{randimen()
```

```
    }\GuIT*}\hfill\vfill}\end{minipage
    }}
68  \begin{document}
69  \pagecolor{black!5!yellow!30}
70  \directlua{maketable()}
71  \noindent\begin{tabu} to \textwidth {@
        {}*{\Tcols}X}
72  \input{table}
73  \end{tabu}
74  \end{document}
```

Questa versione, presente nei sorgenti allegati col nome di `giftpaperluaDI.tex`, sembra notevolmente diversa dalla versione realizzata per generare un foglio A3. Discutiamone le differenze.

Per prima cosa dobbiamo far sì che l'inclusione del pacchetto crop (riga 13) sia indipendente dal foglio che specificheremo a geometry. Per farlo, ci basta definire due variabili di tipo dimensione (righe 7 e 8) a cui assegnare come valore le dimensioni del foglio definite in geometry (righe 9 e 10) aumentate del valore dell'abbondanza (righe 11 e 12) e usare queste variabili nell'includere crop. Quindi bisogna definire due contatori per il numero di riga e di colonna. Nella versione "statica" avevamo scritto questi valori in due variabili locali di Lua. In questo caso non possiamo perché non è consentito usare una funzione Lua nel preambolo, dunque non possiamo comunicare tali valori a LaTeX. Il passaggio attraverso `\newcount` (righe 14–17) è obbligatorio per poter rendere accessibili questi valori alle funzioni Lua senza doverli duplicare esplicitamente nel sorgente. Ora cambia la sintassi d'uso del contatore nell'intestazione della tabella (riga 71), ma è irrilevante: a noi interessa sapere di dover modificare la sola riga 6 per cambiare il foglio e modificare le sole righe 15 e 17 per modificare il numero di righe e di colonne da mettere nella tabella. Ovviamente, poiché le dimensioni delle celle saranno ricalcolate a ogni compilazione, ci serviranno due funzioni Lua che eseguano questo calcolo. Tali funzioni, `pwidth()` (righe 36–38) e `pheight()` (righe 39–41), non fanno altro che emettere rispettivamente un valore dato dal rapporto tra la larghezza del foglio e il numero di colonne della tabella e tra l'altezza del foglio e il numero delle righe della tabella. Questo numero è adimensionale, ma a noi serve un numero con una dimensione ed è per questo che posponiamo a questi valori l'unità di misura (`sp`, lo *scaled point* di TeX, la misura da cui vengono derivate tutte le altre) tramite l'operatore di concatenazione (`..`). Il comando `\cell` (riga 67) richiamerà le due funzioni al posto dei valori *hard coded* della versione "statica".

In accordo con quanto visto per la versione LuaLaTeX, la versione LaTeX (inclusa nei sorgenti come `giftpaperlatexDI.tex`) diventa così:

```
1   \documentclass[12pt]{article}
2
3   \usepackage[T1]{fontenc}
4   \usepackage{guit}
5   \usepackage{tabu}
6   \usepackage[a3paper,hmargin=0pt,
        vmargin=0pt,landscape]{geometry}
7   \newdimen\cropwidth
8   \newdimen\cropheight
9   \cropwidth=\paperwidth
10  \cropheight=\paperheight
11  \advance\cropwidth by 4mm
12  \advance\cropheight by 4mm
13  \usepackage[center,width=\the\
        cropwidth,height=\the\cropheight]{
        crop}
14  \newcounter{Trows}\setcounter{Trows
        }{16}
15  \newcounter{Tcols}\setcounter{Tcols
        }{20}
16  \usepackage{xcolor}
17  \usepackage{graphicx}
18  \usepackage{lcg}
19
20  \makeatletter
21  \def\nloop#1{%
22    \def\nl@@p##1##2\repeat#1{%
23    \def##1{##2\relax\expandafter##1\fi}
          %
24     ##1\let##1\relax}%
25  \expandafter\nl@@p\csname nl@@p-\
        string#1\endcsname
26  }
27  \makeatother
28  \newcommand\dimension[1]{%
29    \ifcase #1\relax
30    \or \large
31    \or \Large
32    \or \LARGE
33    \or \huge
34    \or \Huge
35    \fi}
36  \newcounter{x}
37  \newcommand\cell{\begin{minipage}{\
        cellwidth}\vbox to \cellheight{\
        vfill\hfil\chgrand[first=0, last
        =7] \setbox1=\hbox{\rand}\hskip-\
        wd1\multiply\value{rand} by 45\
        value{x}=\value{rand} \rotatebox[
        origin=c]{\value {x}}{\chgrand[
        first=1, last=5]\setbox1=\hbox{\
        rand}\hskip-\wd1\dimension{\value{
        rand}}\GuIT*}\hfill\vfill} \end{
        minipage}}
38
39  \begin{document}
40  \newwrite\tempfile
41  \immediate\openout\tempfile=table.tex
42  \pagecolor{black!5!yellow!30}
43  \newdimen\cellwidth
44  \newdimen\cellheight
45  \setlength\cellwidth{\paperwidth}
```

```
\setlength\cellheight{\paperheight}
\divide\cellwidth by \theTcols
\divide\cellheight by \theTrows
\newcounter{nrows}
\newcounter{ncols}
\value{nrows}=0
\nloop\a
  \value{ncols}=0
  \nloop\b
    \ifodd\value{nrows}%
      \ifodd\value{ncols}\immediate\
          write\tempfile{\unexpanded{\
          cell}}\fi
    \else%
      \ifodd\value{ncols}\relax\else\
          immediate\write \tempfile{\
          unexpanded{\cell}}\fi
    \fi
    \stepcounter{ncols}
    \ifnum\value{ncols}<\value {Tcols
        }\immediate\write \tempfile{ &}
        %
  \repeat\b
    \immediate\write\tempfile {\
        unexpanded{\\}}\stepcounter{
        nrows}
  \ifnum\value{nrows}<\value{Trows}%
\repeat\a
\immediate\closeout\tempfile
\noindent\begin{tabu} to \textwidth {@
    {}*{\theTcols}X}
\input{table}
\end{tabu}
\end{document}
```

Le righe 43–48 mostrano il codice LaTeX per il calcolo delle dimensioni di ogni cella. Le dimensioni `\cellwidth` e `\cellheight`, usate nel comando `\cell`, dovranno contenere le dimensioni orizzontale e verticale della cella. Dopo averle create e inizializzate alla larghezza e all'altezza della carta, le dividiamo rispettivamente per il numero di colonne (celle orizzontali) e di righe (celle verticali). Queste ultime quantità le avevamo definite alle righe 14–15. Dunque, quando decideremo di far compilare un documento su un foglio diverso dall'A3 scritto nel sorgente, o con righe e/o colonne in quantità differenti, tutti i cambiamenti che faremo sono concentrati alle righe 6 (formato della carta) e 14–15 (numero di righe e colonne).

Infine la versione LaTeX3 (`giftpaper3DI.tex`):

```
\documentclass[12pt]{article}

\usepackage[T1]{fontenc}
\usepackage{guit}
\usepackage{tabu}
\usepackage[a3paper,hmargin=0pt,
    vmargin=0pt,landscape]{geometry}
\usepackage{xcolor}
\usepackage{graphicx}
\usepackage{expl3}

\ExplSyntaxOn
\dim_new:N\cropwidth\dim_set:Nn\
    cropwidth{\dim_eval:n{\paperwidth
    + 4mm}}
\dim_new:N\cropheight\dim_set:Nn\
    cropheight{\dim_eval:n{\
    paperheight + 4mm}}
\usepackage[center,width=\the\
    cropwidth,height=\the\cropheight]{
    crop}
\int_new:N\Trows\int_set:Nn\Trows{16}
\int_new:N\Tcols\int_set:Nn\Tcols{20}

\cs_new:Npn \dimension #1
{ \if_case:w #1 \relax
    \or: \large
    \or: \Large
    \or: \LARGE
    \or: \huge
    \or: \Huge
  \fi}
\cs_new:Npn \cell
{\begin{minipage}{\cellwidth}\vbox to
    \cellheight{\vfill \hfil\rotatebox
    [origin=c]{\fp_eval:n {45*randint
    (0,7)}}{\dimension{\fp_eval:n{
    randint(1,5)}}\GuIT*}\hfill\vfill
    }\end{minipage}}
\ExplSyntaxOff
\begin{document}
\pagecolor{black!5!yellow!30}
\ExplSyntaxOn
\iow_new:N \tempfile
\iow_open:Nn \tempfile {table.tex}
\dim_new:N\cellwidth\dim_set:Nn\
    cellwidth{\dim_eval:n{\paperwidth
    /\Tcols}}
\dim_new:N\cellheight\dim_set:Nn\
    cellheight{\dim_eval:nv{\
    paperheight/\Trows}}
\int_new:N\nrows\int_set:Nn\nrows{0}
\int_new:N\ncols
\int_do_while:nNnn{\nrows}<{\Trows}{%
  \int_set:Nn\ncols{0}
  \int_do_while:nNnn{\ncols}<{\Tcols}{
      %
    \int_if_odd:nTF{\nrows}{%
      \int_if_odd:nTF{\ncols}{%
        \iow_now:Nn \tempfile {\cell}
      }{\relax}
    }{%
       \int_if_odd:nTF\ncols{\relax}{
           %
         \iow_now:Nn \tempfile {\cell}
       }%
    }
    \int_incr:N\ncols
```

```
51      \int_compare:nNnTF{\ncols}< {\
            Tcols}{%
52        \iow_now:Nn \tempfile{ &}
53      }{\relax}
54    }
55    \iow_now:Nn \tempfile {\\}
56    \int_incr:N\nrows
57  }
58  \iow_close:N \tempfile
59  \noindent\begin{tabu} to \textwidth {@
        {}*{\Tcols}X}
60  \input{table}
61  \end{tabu}
62  \ExplSyntaxOff
63  \end{document}
```

Come già detto per la versione "statica", anche questa versione è la pura traduzione del programma equivalente scritto in LaTeX, quindi non ripeterò la spiegazione che sarebbe una ripetizione. Come molti utenti di LaTeX3, posso senz'altro affermare, di aver apprezzato molto la coerenza delle interfacce delle funzioni di questa versione del linguaggio e, di conseguenza, la sua facilità d'uso.

10 Conclusioni e proposta

Il presente articolo ha mostrato alcuni metodi per ottenere della carta da regalo personalizzata con del testo o con un'immagine a piacere. I metodi mostrati sono eterogenei: C + LaTeX, LuaLaTeX, LaTeX e LaTeX3 in rappresentanza di tre filosofie: calcolare all'esterno tutti i valori da inserire in un documento LaTeX tramite un linguaggio non integrato (C) o integrato (LuaTeX), oppure calcolare detti valori internamente grazie a LaTeX(3). Ogni lettore può ricorrere alla tecnica che meglio gli si confà.

La proposta al Direttivo è quella di considerare l'adozione della carta da regalo personalizzata GUIT come gadget di benvenuto ai nuovi soci.

A Ricerche ottimizzatorie e perdite di tempo

La sezione 3 ha mostrato una possibile soluzione (contenuta nel file `makeGPtv1.c`) per risolvere il problema della produzione della carta regalo e, al margine, una possibile soluzione al riempimento a scacchiera di una tabella. Come sempre, ogni soluzione ha vantaggi e svantaggi. Per esempio, in passato non sempre il codice più leggibile era il più efficiente. Ora, cercare di ottimizzare un programma così banale è inutile: quante volte potrà mai essere eseguito? E quanto tempo occorre per un'esecuzione? Quindi, se pure si guadagnassero ben 2 ms, quale sarebbe il vantaggio? Specie se l'ottimizzazione porta via troppo tempo. Tuttavia tentiamo un'impossibile miglioria tenendo conto che i compilatori odierni sono talmente efficienti da rendere inutili le riscritture del codice più risapute.

Tabella 1: Rilevazione dei tempi medî di esecuzione di `makeGPtv1` (sezione 3).

Programma	Tempo utente (s)	Tempo totale (s)	CPU (%)
`time`[10]	0,242	0,242	—
`/usr/bin/time`[11]	0,235	0,236	99,6
`runtime`[12]	0,24	0,24	99

Nella trattazione che segue sono stati usati i seguenti programmi per rilevare i tempi: `time`, comando integrato di bash tanto da esserne parola riservata; `time`, la versione GNU del comando; `runtime`, utility descritta in Peek *et al.* (1997). Tutti i tempi sono stati rilevati sulla stessa macchina (Intel Core i7 con 16 GB di RAM) e nelle stesse condizioni (due terminali e `xpdf` attivi; su un terminale si rilevano i tempi di esecuzione, sull'altro, terminate le rilevazioni, si lancia `vi` per riportare i tempi rilevati). L'eseguibile da testare è stato preparato eliminando le righe di codice 35–40, cioè quelle relative alla scrittura della tabella su file e di compilazione del documento LaTeX. Dunque ci si concentra solo sull'efficienza del riempimento della tabella in memoria. L'esecuzione del file C così preparato dà come tempo di esecuzione zero, cioè inferiore al millisecondo. Per rendere più corpose le misure, il sorgente è stato modificato introducendovi un ciclo che ripete 10 000 volte le righe 22–34. I tempi sono quelli medî relativi a 25 esecuzioni del programma. La tabella 1 riporta i dati di esecuzione forniti dai tre "cronometri".

Si può notare che i tempi rilevati dai comandi di shell (`time` di bash e `runtime`, basato su csh e che misura i tempi singoli al millesimo ma dà la media al centesimo) sono coerenti tra loro, mentre quello di `time` di GNU dà risultati inferiori fino a un centesimo di secondo. Su `https://unix.stackexchange.com/questions/27920/why-bash-time-is-more-precise-then-gnu-time` si trova la seguente risposta:

> After some hardcore bash code examining I found out that bash `time`

10. Il comando fornisce i tempi fino al millisecondo e la media delle esecuzioni, calcolata con un programma esterno, è stata arrotondata alla terza cifra decimale. Non dà percentuali di CPU.

11. Il comando fornisce i tempi fino al centesimo e la media delle esecuzioni, calcolata con un programma esterno, è stata arrotondata comunque alla terza cifra decimale per avere un'idea del bilanciamento dei tempi di esecuzione. La percentuale di CPU è invece data senza decimali e il suo valor medio è stato arrotondato alla prima cifra decimale.

12. Lo script `runtime` sfrutta gli strumenti di csh per le rilevazioni e i calcoli. Lo *user time* rilevato arriva ai millesimi di secondo, ma il calcolo della media viene arrotondato ai centesimi di secondo. Il *total time* (user + system) è rilevato ed espresso in minuti + secondi + centesimi ma il valore medio riporta solo i minuti e i secondi. Infine, la percentuale di CPU è rilevata al primo decimale ma il calcolo del valor medio viene espresso senza decimali. Questa e le successive tabelle riporteranno il *total time*, pari allo *user time* quando il *system time* è 0, e la percentuale di CPU rilevata dallo script.

> uses `getrusage()` and GNU `time` uses `times()`. `getrusage()` is far more precise because of microsecond resolution.

Il perché di questa differenza di misurazione può probabilmente essere imputato all'effetto dei troncamenti dei valori rilevati. Neanche le informazioni riportate su https://stackoverflow.com/questions/12392278/measure-time-in-linux-time-vs-clock-vs-getrusage-vs-clock-gettime-vs-gettimeof attestano con chiarezza il motivo di tale differenza. Non essendo questo un argomento strettamente attinente al lavoro presentato, lascerò a quei lettori che abbiano necessità di farlo il compito di scoprire i motivi delle differenze di misurazione. Se tali lettori lo riterranno opportuno, potranno comunicare l'esito dei loro approfondimenti direttamente a me o in una futura comunicazione su ArsTEXnica.

Si possono migliorare le prestazioni della versione in C del programma? Ci si può certamente provare, sempre ricordando che stiamo tentando di limare millesimi a una versione che esegue 10 000 volte la stessa cosa, il che ci farà limare un decimillesimo di quei millesimi nel programma reale. Dunque, affrontiamo il compito con puro spirito didattico.

In prima istanza si può pensare che togliere l'operazione modulo (`%`) dal ciclo alla riga 29 possa aiutare. Per farlo, basta applicare la strategia adottata da due degli autori di Beccari *et al.* (2016): sdoppiare il ciclo per operare sulle righe dispari indipendentemente da quelle pari. Dunque, le righe 28–34 saranno sostituite da:

```
28    for (i = 0; i < NROWS; i += 2)
29      for (j = 0; j < NCOLS; j += 2) {
30        rotation = ((int) rand() %
              MAXROTSTEP ) * DEGSTEP;
31        dimpos = (int) rand() % NUMDIM;
32        sprintf (tmp, "\\begin{minipage
              }{2.1cm}\\vbox␣to␣1.856cm{\\
              vfill\\hfil\\rotatebox[
              origin=c]{%d}{%s\\GuIT*}\\
              hfill\\vfill}\\end{minipage}
              ␣%s", rotation, dimen[dimpos
              ], tabular[i][j]);
33        strcpy (tabular[i][j], tmp);
34      }
35    for (i = 1; i < NROWS; i += 2)
36      for (j = 1; j < NCOLS; j += 2) {
37        rotation = ((int) rand() %
              MAXROTSTEP ) * DEGSTEP;
38        dimpos = (int) rand() % NUMDIM;
39        sprintf (tmp, "\\begin{minipage
              }{2.1cm}\\vbox␣to␣1.856cm{\\
              vfill\\hfil\\rotatebox[
              origin=c]{%d}{%s\\GuIT*}\\
              hfill\\vfill}\\end{minipage}
              ␣%s", rotation, dimen[dimpos
              ], tabular[i][j]);
```

Tabella 2: Rilevazione dei tempi medî di esecuzione di `makeGPtv2` (appendice A).

Programma	Tempo utente (s)	Tempo totale (s)	cpu (%)
`time`	0,242	0,242	—
`/usr/bin/time`	0,233	0,233	99,5
`runtime`	0,24	0,24	99

```
40        strcpy (tabular[i][j], tmp);
41      }
```

Nei sorgenti allegati, il file in questione è `makeGPtv2.c` (ovviamente nella versione con scrittura della tabella su file e senza i 10 000 cicli inutili). Il doppio ciclo delle righe 28–34 si occupa di riempire le celle pari (0, 2, 4...) delle righe pari (0, 2, 4...); quello delle righe 35–41 riempie invece le celle dispari (1, 3, 5...) delle righe dispari (1, 3, 5...). Il codice è un po' più lungo e forse un po' più chiaro del precedente; il vantaggio nell'esecuzione (preparata con gli stessi criterî già descritti) è invece nullo, come si evince dalla tabella 2. Ciò induce a pensare che il compilatore esegua in autonomia lo sdoppiamento del ciclo.

Possiamo fare un altro tentativo di migliorare le prestazioni tenendo conto del fatto che il C memorizza gli array per riga in posizioni consecutive di memoria: tutti i dati della seconda riga di un array occuperanno le posizioni di memoria immediatamente successive a tutti i dati della prima riga dello stesso array. Questo, a costo di un po' di aritmetica dei puntatori aggiuntiva, permette di accedere a ogni posizione dell'array sommando gli indici all'indirizzo base dell'array anziché tramite gli operatori parentesi quadre. Lasciando inalterato il doppio ciclo (righe 28–34) della versione originale, modifichiamo solo le due righe seguenti:

```
32        sprintf (tmp, "\\begin{minipage
              }{2.1cm}\\vbox␣to␣1.856cm{\\
              vfill\\hfil\\rotatebox[
              origin=c]{%d}{%s\\GuIT*}\\
              hfill\\vfill}\\end{minipage}
              ␣%s", rotation, dimen[dimpos
              ], *(*(tabular + i) + j));
33        strcpy (*(*(tabular + i) + j),
              tmp);
```

che, come vediamo, sono meno comprensibili da chi abbia dimestichezza con gli accessi a un array solo tramite operatori parentesi quadre.

Tale modifica (file `makeGPtv3`) sembra essere "efficace" poiché i tempi di esecuzione sono leggeremente inferiori (tabella 3) rispetto agli altri casi.

Come ulteriore e ultimo tentativo, possiamo unificare il doppio ciclo. Come vedremo tra poco, questo comporta dei controlli un po' meno intuitivi per capire quando ci si trova su una riga pari o su una dispari. Innanzitutto ci serviranno tre ulteriori variabili di tipo `int`: le chiameremo `cells`, `rows`

Tabella 3: Rilevazione dei tempi medî di esecuzione di `makeGPtv3` (appendice A).

Programma	Tempo utente (s)	Tempo totale (s)	cpu (%)
`time`	0,239	0,24	—
`/usr/bin/time`	0,232	0,233	99,5
`runtime`	0,24	0,24	99

Tabella 4: Rilevazione dei tempi medî di esecuzione di `makeGPtv4` (appendice A).

Programma	Tempo utente (s)	Tempo totale (s)	cpu (%)
`time`	0,246	0,246	—
`/usr/bin/time`	0,242	0,243	99,6
`runtime`	0,24	0,24	99

e `cols`. Poi dobbiamo modificare dalla riga 28 del programma originale:

```
28    cells = NROWS * NCOLS;
29    for (i = 0; i < cells; i += 2) {
30      rotation = ((int) rand() %
            MAXROTSTEP ) * DEGSTEP;
31      dimpos = (int) rand() % NUMDIM;
32      sprintf (tmp, "\\begin{minipage
            }{2.1cm}\\vbox␣to␣1.856cm{\\
            vfill\\hfil\\rotatebox[origin=
            c]{%d}{%s\\GuIT*}\\hfill\\
            vfill}\\end{minipage}␣%s",
            rotation, dimen[dimpos], *(*(
            tabular) + i));
33      strcpy (*(*(tabular) + i), tmp);
34      rows = i / NCOLS;
35      cols = i % NCOLS;
36      if ((cols + 2) >= NCOLS) {
37        if (rows % 2) i--;
38        else i++;
39      }
40    }
```

Vediamo il funzionamento di questa versione (`makeGPtv4.c` nei sorgenti): calcola il numero di celle della tabella ed esegue un unico ciclo su tutte le celle saltandone una ogni due. Genera, come sempre, la rotazione e la dimensione, quindi memorizza la stringa LaTeX nella cella individuata da `*(*tabular) + i`, cioè la posizione iniziale in memoria dell'array aumentata del numero di righe per il numero di colonne (numero memorizzato in `i`) per il numero di byte della cella. Quest'ultima moltiplicazione è "gratuita" nell'aritmetica dei puntatori perché implicita nel tipo di variabile e dunque è calcolato direttamente dal C, come anche nel sorgente precedente in cui però gli indici rimanevano sdoppiati. Ora bisogna capire se siamo all'inizio di una riga e se questa è pari o dispari e avere un solo indice non ci aiuta. Dunque dobbiamo calcolare la riga e la colonna correnti a partire dall'indice `i` (righe 34–35) e chiederci se siamo prossimi alla nuova riga (l'ultima colonna di una riga è la 19ª; la prima iterazione arriverà alla 18ª prima di andare alla riga successiva, quindi bisogna chiedersi se il numero di colonna corrente +2 raggiunge o eccede il numero di colonne, nel qual caso si controlla se la riga è pari (e quindi ci si sposta in avanti di una cella) o dispari (caso per cui ci si sposta indietro di una cella). Non dimentichiamo mai che la prima riga e la prima colonna hanno indice 0 e sono considerate pari.

I maggiori controlli e calcoli di questa versione si riflettono sull'esecuzione nell'ordine dei millesimi (tabella 4). Pur nell'esiguità dell'aumento dei tempi, il buon senso sconsiglia di ricorrere a una tale soluzione anche per problemi di leggibilità e manutenibilità. Tutti questi esercizî di programmazione sono serviti a capire il vantaggio di un approccio rispetto agli altri e a rivedere alcuni concetti del linguaggio C o di ottimizzazione del codice (per esempio il *loop splitting* menzionato anche in Falk e Marwedel (2004, 2003)).

Allora ci arrendiamo qui? No, però ricorriamo alla forza bruta. Il compilatore usato in questi test è lo `gnu gcc`. Questo consente diversi livelli di ottimizzazione del codice. Col livello 3 o col livello *fast* si ottiene un tempo medio di esecuzione (calcolato da `runtime`) di 23 centesimi di secondo, cioè un centesimo in meno dei tempi precedenti.

Ciò conferma definitivamente quello che avevamo già detto all'inizio e spesso ripetuto: inutile tentare di migliorare il codice originale il cui numero di esecuzioni e tempo di esecuzione sono e saranno sempre esigui.

B Un confronto grezzo

Arrivati alla fine, e oltre, dell'articolo, come ci si può orientare verso la migliore soluzione? Stante che una soluzione migliore in assoluto non c'è, possiamo però fare alcune considerazioni per orientarci singolarmente verso una soluzione piuttosto che verso un'altra.

Prendiamo in considerazione alcuni fattori per ognuno dei quattro programmi da confrontare; ogni fattore orienterà un tipo di decisione: quante righe di codice compongono ogni programma?[13] Quanti linguaggi bisogna conoscere?[14] Quanti fi-

13. Il numero di righe di codice può non essere indicativo, perché in alcuni casi due o più comandi o istruzioni sono stati posti su un'unica riga e, a parte casi speciali, potremmo decidere di scrivere il programma su un'unica, lunga riga. Forse sarebbe più significativo il numero di comandi o istruzioni date. Ma diamo questo dato solo come "indice di tempo di stesura del programma", senza contare il tempo speso a ideare l'algoritmo e a codificarlo.

14. Questo dato è significativo soprattutto per chi non abbia interesse o necessità di conoscere linguaggi diversi da LaTeX.

le formano il programma?[15] Quant'è il tempo di esecuzione?[16]

La tabella 5 riassume tutti questi numeri. Il lettore ha la più completa libertà di interpretarli per capire se la soluzione che avrebbe scelto è veramente la migliore (per lui).

Come promesso, non discuterò i dati riportati nella tabella 5 ma vorrei porre l'attenzione mia e dei lettori su un dato apparentemente sconcertante: la versione C + LaTeX, pur lanciando una compilazione con LaTeX, risulta molto più veloce della compilazione diretta con LaTeX. La prima spiegazione che mi viene in mente è che, mentre la tabella generata dal programma C mette una riga della tabella su una riga del file, la tabella generata dal documento LaTeX mette su ogni riga del file una singola cella. Dunque la lettura del file-tabella è molto più lenta. Ma non dobbiamo dimenticare che anche i numeri casuali sono generati da un pacchetto di LaTeX e dalla sua aritmetica durante la compilazione del documento anziché da una libreria del C altamente efficiente prima che il documento sia compilato.

Se qualcuno, poi, si sorprenderà per la prestazione temporale di LuaLaTeX, potrà riflettere sul fatto che Lua è un linguaggio interpretato e non compilato.

C La versione di Egregio

Alla fine della scrittura dell'articolo, ho pensato di far leggere il frutto delle mie "fatiche" proprio all'Egregio Guru nel timore che le mie citazioni "spiritaffettuose" potessero risultare indisponenti. Non ho ricevuto alcuna osservazione in merito, ma in compenso ho ricevuto una lezione di programmazione senza pari.

Oltre ad avermi dato alcuni consigli per impaginare meglio l'articolo, mi ha mandato una versione "statica" in LaTeX3 che, pur "tradendo" il codice originale nel più puro spirito delle traduzioni, fa delle cose che solo una profonda conoscenza di TeX e una conoscenza di frontiera di LaTeX3[17] permettono di fare: riempie la tabella senza cicli `while` ma con le funzioni dedicate, e lo fa senza scrivere alcunché su file esterni. In più, usa una versione di `\int_case` che permette di specificare i numeri per il confronto, per cui permette un'espansione più facile e intuitiva verso eventuali dimensioni del font più piccole di `\normalsize`. Bisogna ammettere che l'espressività del codice dell'Egregio Guru, paragonata all'espressività del mio, è la stessa di uno scritto di Umberto Eco paragonato a una pagina di astine (o un discorso di Woodstock con Snoopy).

15. Oltre che sul numero totale di righe di codice, questo fattore incide su come mettere in cllegamento i varî file. Non fanno parte del dato i file generati automaticamente, ma solo quelli scritti a mano.

16. Quello qui riportato è il tempo medio su 25 esecuzioni misurate col `time` di bash.

17. Non potete compilare questo documento con una TeX Live 2018 o precedente.

Riporto qui di seguito il codice, così come mi è arrivato (e incluso nei sorgenti come `giftpaper3egreg.tex`): tutti faremo bene a studiarlo nei minimi dettagli.[18]

```
\documentclass[12pt]{article}

\usepackage[T1]{fontenc}
\usepackage{guit}
\usepackage[a3paper,hmargin=0pt,
    vmargin=0pt,landscape]{geometry}
\usepackage[center,width=42.4cm,height
    =30.1cm]{crop}
\usepackage{xcolor}
\usepackage{graphicx}
\usepackage{expl3}

\ExplSyntaxOn
\cs_new_protected:Npn \cartaregalo_
    size:n #1
{
 \int_case:nn { #1 }
  {
   {1}{\large}
   {2}{\Large}
   {3}{\LARGE}
   {4}{\huge}
   {5}{\Huge}
  }
}
\cs_new_protected:Npn \cartaregalo_
    cell:
{
 \begin{minipage}{2.1cm}
 \leavevmode\vbox to 1.856cm
  {
   \vss
   \hbox to 2.1cm
    {
     \hss
     \rotatebox[origin=c]{\fp_eval:n
         {45*randint(0,7)}}
      {
       \cartaregalo_size:n {\fp_eval:n
           {randint(1,5)}} \GuIT*
      }
     \hss
    }
   \vss
  }
 \end{minipage}
}
\ExplSyntaxOff

\begin{document}
\pagecolor{black!5!yellow!30}
```

18. Posso affermare che la visione di tale sorgente, ispirandomi una citazione cólta, mi ha riportato alla mente un apoftegma attribuito ad Alvaro Vitali: «davanti all'arte togliti le mutande e mettile da parte» (Aa.Vv., 1996, p. 74).

Tabella 5: Confronto numerico dei quattro programmi "statici".

Programma	Nº file	Nº Linguaggi	Righe di codice	Tempo di esecuzione (s)
C + LaTeX	2	2	51 + 17	0,354
LuaLaTeX	1	2	56	0,977
LaTeX	1	1	58	0,549
LaTeX3	1	2	59	0,696

```
\ExplSyntaxOn
\tl_new:N \l_cartaregalo_body_tl
\int_const:Nn \c_cartaregalo_trows_int
     { 16 }
\int_const:Nn \c_cartaregalo_tcols_int
     { 20 }
\int_step_inline:nn { \c_cartaregalo_
    trows_int }
{
 \int_step_inline:nn { \c_cartaregalo_
     tcols_int }
  {
   \int_if_odd:nTF { #1 }
    {
     \int_if_odd:nT { ##1 }
      {
       \tl_put_right:Nn \l_cartaregalo
           _body_tl { \cartaregalo_cell
           : }
      }
    }
    {
     \int_if_odd:nF { ##1 }
      {
       \tl_put_right:Nn \l_cartaregalo
           _body_tl { \cartaregalo_cell
           : }
      }
    }
   \int_compare:nT { ##1 < \c_
       cartaregalo_tcols_int }
    {
     \tl_put_right:Nn \l_cartaregalo_
         body_tl { & }
    }
  }
 \tl_put_right:Nn \l_cartaregalo_body_
     tl { \\ }
}
\setlength{\tabcolsep}{0pt}
\noindent\begin{tabular*}{\textwidth}{
    @{\extracolsep{\fill}}*{20}{c}@{}}
\tl_use:N\l_cartaregalo_body_tl
\end{tabular*}
\ExplSyntaxOff
\end{document}
```

Per un giusto confronto, il tempo medio di esecuzione del codice (sempre su 25 esecuzioni) è risultato 0,36 s: veramente un tempo notevole.

Ringraziamenti

Questo articolo è debitore di alcune persone. Iniziamo da Roberto Giacomelli e Luigi Scarso; il primo attivissimo nel divulgare LuaTeX, il secondo nello svilupparlo. È grazie al loro esempio che ho deciso di fare una "traduzione" del mio programma in LuaLaTeX. Poi Claudio Beccari: non solo l'occasione del regalo mi ha suggerito di preparare della carta da regalo speciale, ma un suo recente messaggio mi ha dato lo spunto finale per scrivere la versione in LaTeX3; non bastando ciò, si è prestato a leggere il draft definitivo dell'articolo a scatola chiusa e mi ha fornito diversi suggerimenti per migliorare tutti i codici, sia dal punto di vista dello sviluppatore, sia da quello dell'utente. Infine, Enrico Gregorio e il suo *alter ego* (inventato ma verosimile) Egregio Guru: lungi dal chiedergli lumi su qualunque cosa inerente LaTeX3, ho preferito studiarmi i suoi articoli e il suo pacchetto kantlipsum; le cocenti esperienze passate mi hanno consentito di prevedere e prevenire un paio di sue possibili osservazioni che ho scritto nell'articolo fingendo che me le abbia fatte lui (ma, per sua ammissione, non mi avrebbe consigliato lcg). Però la realtà è stata più incredibile della fantasia!

Riferimenti bibliografici

Aa.Vv. (1996). «Campagna abbonamenti». *Amarcord. Il lato oscuro del cinema*, (1).

Beccari, Claudio (2019). «Email del 27 e 28 ottobre 2019». Comunicazione privata.

Beccari, Claudio, Roberto Giacomelli e Maurizio Molinaro (2016). «Il concorso della scacchiera». *ArsTEXnica*, (25).

LuaTeX development team (2019). *LuaTeX Reference Manual*. www.luatex.org.

Falk, Heiko e Peter Marwedel (2003). «Control flow driven splitting of loop nests at the source code level». In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1*. IEEE Computer Society, Washington, DC, USA, DATE '03. http://dl.acm.org/citation.cfm?id=789083.1022762.

— (2004). *Source Code Optimization Techniques for Data Flow Dominated Embedded Software*. Kluwer Academic Publishers, Boston.

Giacomelli, Roberto e Gianluca Pignalberi (2015). «Generare documenti LaTeX con diversi linguaggi di programmazione». *ArsTEXnica*, (20).

Kaps, Florian "Doc" (2018). «The magic of linotype: Save and restart a printing legend». https://www.kickstarter.com/projects/1755997589/the-magic-of-linotype-save-and-restart-a-printing/description.

Peek, Jerry, Tim O'Reilly e Mike Loukides (1997). *Unix Power Tools*. O'Reilly, Sebastopol. Reperibile online su https://docstore.mik.ua/orelly/unix/upt/ch39_04.htm (sezione 39-4, Runtime) e ftp://ftp.oreilly.com/published/oreilly/power_tools/unix/upt9707.tgz (sorgente).

Plazzi, Andrea e Edoardo Rosati (a cura di) (1996). *Al servizio dell'Eroe. Il Tex di Magnus*. Editrice PuntoZero, Bologna.

print24 (2019). «Carta regalo». https://print24.com/it/prodotti-di-stampa/carte-da-regalo/. Ultima consultazione: 18 novembre 2019.

Stack Exchange (2015). «Making a random number». https://tex.stackexchange.com/questions/212738/making-a-random-number.

The LaTeX3 Project (2019). *The LaTeX3 Interfaces*. Leggibile dando da terminale il comando `texdoc interface3`.

Il codice illustrato in questo articolo è disponibile, per i soci GUIT, a questo indirizzo: https://www.guitex.org/home/images/ArsTeXnica/codice/

▷ Gianluca Pignalberi
g dot pignalberi at gmail dot com

siunitx: le macro fondamentali e la composizione delle tabelle

*Joseph Wright**

Sommario

Dopo una ricognizione dei comandi principali del pacchetto siunitx, se ne descrivono nel dettaglio le funzionalità specifiche per comporre tabelle di numeri e quantità fisiche.

Abstract

After a survey of the main commands of the siunitx package, we describe in detail its specific functionalities to compose tables of numbers, units and physical quantities.

1 Introduzione

Una *quantità fisica* è una proprietà (di fenomeno, corpo o sostanza) quantificabile con una misurazione e si esprime come il prodotto di un fattore moltiplicativo (un numero) per un'unità (espressa con una o più lettere o con altri simboli): per esempio, 1 mm, 28 °C, 220 V. La loro composizione richiede attenzione per garantire che il significato matematico della combinazione dei due elementi sia chiaro. Nonostante che unità e relative regole per adoperarle siano stabilite in modo inequivocabile dal sistema SI, alcuni Paesi ed editori adottano convenzioni differenti in quanto al loro aspetto tipografico.

Il pacchetto siunitx fornisce una serie di strumenti per comporle in modo coerente e un'ampia scelta di opzioni di configurazione che permettono di seguire le diverse convenzioni tipografiche. Inoltre, elabora automaticamente numeri e quantità fisiche e ne controlla molto finemente l'allineamento nelle tabelle, rispondendo praticamente a ogni esigenza dell'utente. Si potranno così ottenere documenti finali molto flessibili, senza dover modificare in modo sostanziale la sintassi nel testo sorgente.

2 siunitx per l'impaziente

Il pacchetto fornisce all'utente le macro e gli specificatori generali mostrati nella tabella 1.

Per impostazione predefinita, tutto il testo negli argomenti delle macro è composto nel font matematico diritto corrente. Lo si può avere nel font impostato per il documento passando al pacchetto l'opzione `detect-all`.

Secondo le convenzioni in vigore nel nostro Paese, negli esempi di questo articolo il separatore decimale è la virgola e gli ultimi elementi delle liste sono preceduti da congiunzioni e preposizioni italiane. Per ottenere questo risultato, vanno passate al pacchetto le seguenti impostazioni, che sovrascrivono quelle predefinite per la lingua inglese:

```
\sisetup{
 output-decimal-marker = {,},
 list-final-separator  = { \translate{e} },
 list-pair-separator   = { \translate{e} },
 range-phrase          = { \translate{a} }
}
```

Naturalmente, nell'argomento di `\translate` si può mettere un testo a piacere.

È importante notare che tutte le macro descritte in questo articolo funzionano sia in modo matematico sia in modo testuale.

3 Uso di siunitx

Caricamento

Il pacchetto si carica nel modo consueto:

```
\usepackage{siunitx}
```

ed è retrocompatibile con la versione precedente scrivendo

```
\usepackage[version-1-compatibility]%
  {siunitx}
```

Questa funzionalità è utile per chi avesse maggiore dimestichezza con le vecchie macro o dovesse lavorare a documenti datati.

Opzioni

Il comportamento di siunitx è governato da una serie di ⟨*opzioni*⟩ disponibili nella forma tradizionale o nel tipo ⟨*chiave*⟩=⟨*valore*⟩, da passare al pacchetto come segue:

```
% nel preambolo del documento
\usepackage[⟨opzioni⟩]{siunitx}

% nel preambolo e nel corpo del documento
\sisetup{⟨opzioni⟩}
```

*Il contenuto di questo articolo è una selezione dalla documentazione del pacchetto siunitx, tradotta da Tommaso Gordini con il permesso dell'autore. La responsabilità di eventuali errori o fraintendimenti del testo originale è del traduttore, il quale desidera ringraziare Enrico Gregorio per i suoi preziosi chiarimenti. (*N.d.T.*)

Tabella 1: Macro fondamentali definite da siunitx.

La macro...	...compone
`\sisetup{`⟨*opzioni*⟩`}`	
`\num[`⟨*opzioni*⟩`]{`⟨*numero*⟩`}`	numeri
`\numlist[`⟨*opzioni*⟩`]{`⟨*numeri*⟩`}`	liste di numeri
`\numrange[`⟨*opzioni*⟩`]{`⟨*numero*$_1$⟩`}{`⟨*numero*$_2$⟩`}`	intervalli numerici
`\ang[`⟨*opzioni*⟩`]{`⟨*angolo*⟩`}`	angoli
`\si[`⟨*opzioni*⟩`]{`⟨*unità*⟩`}`	unità
`\SI[`⟨*opzioni*⟩`]{`⟨*numero*⟩`}[`⟨*preunità*⟩`]{`⟨*unità*⟩`}`	quantità fisiche
`\SIlist[`⟨*opzioni*⟩`]{`⟨*numeri*⟩`}{`⟨*unità*⟩`}`	liste di quantità fisiche
`\SIrange[`⟨*opzioni*⟩`]{`⟨*numero*$_1$⟩`}{`⟨*numero*$_2$⟩`}{`⟨*unità*⟩`}`	intervalli di quantità fisiche
`\tablenum[`⟨*opzioni*⟩`]{`⟨*numero*⟩`}`	allineamenti complessi
`S[`⟨*opzioni*⟩`]`	colonne di numeri
`s[`⟨*opzioni*⟩`]`	colonne di unità

Si noti che il comando `\sisetup` modifica localmente le opzioni globali solo se dato *dentro* un ambiente.

4 Numeri

Numeri puri

Per comporre i numeri c'è il comando

```
\num[⟨opzioni⟩]{⟨numero⟩}
```

che formatta automaticamente il ⟨*numero*⟩ rimuovendo gli eventuali spazi, riconoscendo gli esponenti e aggiungendo le spaziature appropriate prima e dopo il separatore decimale e nei numeri lunghi. Per impostazione predefinita, se necessario viene aggiunto uno zero prima del separatore decimale, che nel testo sorgente può essere '`.`' o '`,`' indifferentemente.

```
\num{123}          \\
\num{1234}         \\
\num{12345}        \\
\num{0,123}        \\
\num{0,1234}       \\
\num{,12345}       \\
\num{12345,67890}
```

123
1234
12 345
0,123
0,1234
0,123 45
12 345,678 90

Vengono riconosciuti anche esponenti (indicati con uno dei caratteri `e`, `E`, `d`, `D`), numeri complessi e segni di moltiplicazione.

```
\num{,3e45}    \\
\num{3,45d-4}  \\
\num{-e10}     \\
\num{1+-2i}    \\
\num{9 x 2,3 x 5,4}
```

$0{,}3 \times 10^{45}$
$3{,}45 \times 10^{-4}$
-10^{10}
$1 \pm 2\mathrm{i}$
$9 \times 2{,}3 \times 5{,}4$

Liste di numeri

Per comporre liste di numeri c'è il comando

```
\numlist[⟨opzioni⟩]{⟨numeri⟩}
```

I ⟨*numeri*⟩ della lista vanno separati con il punto e virgola e messi in un gruppo, dato che la lunghezza della lista è flessibile. Il comando aggiunge del testo tra penultimo e ultimo elemento della lista.

```
\numlist{7;30;50} \\
\numlist{10;30}
```

7, 30 e 50
10 e 30

Intervalli di numeri

Per comporre intervalli numerici c'è il comando

```
\numrange[⟨opzioni⟩]{⟨numero1⟩}{⟨numero2⟩}
```

che funziona come `\numlist`, ma aggiunge del testo tra i due estremi dell'intervallo.

```
\numrange{10}{30}
```

10 a 30

5 Angoli

Per comporre gli angoli c'è il comando

```
\ang[⟨opzioni⟩]{⟨angolo⟩}
```

L'⟨*angolo*⟩ può essere scritto sia come numero decimale sia come una lista di gradi, minuti e secondi separati con il punto e virgola. I numeri che indicano gli angoli sono formattati esattamente come gli altri numeri.

```
\ang{10}       \\
\ang{12.3}     \\
\ang{4,5}      \\
\ang{1;2;3}    \\
\ang{13;;}     \\
\ang{;7;}      \\
\ang{;;1}      \\
\ang{+10;;}    \\
\ang{-0;1;}
```

$10°$
$12{,}3°$
$4{,}5°$
$1°2'3''$
$13°$
$7'$
$1''$
$10°$
$-0°1'$

6 Unità pure

Per comporre il simbolo di un'unità c'è il comando

```
\si[⟨opzioni⟩]{⟨unità⟩}
```

che accetta le ⟨*unità*⟩ scritte in due modi diversi.

Quando l'⟨*unità*⟩ contiene lettere o numeri da usare direttamente, siunitx converte '`.`' e '`~`' in opportuno 'materiale interunità' e colloca correttamente gli esponenti e i deponenti specificati con '`^`' e '`_`'.

```
\si{kg.m.s^{-1}} \\
\si{kg.m/s^2}    \\
\si{g_{pol}~mol_%
  {cat}.s^{-1}}
```

$\mathrm{kg\,m\,s^{-1}}$
$\mathrm{kg\,m/s^{2}}$
$\mathrm{g_{pol}\,mol_{cat}\,s^{-1}}$

Il secondo modo richiede di inserire unità, prefissi multipli SI e potenze tramite apposite macro, l'elenco completo delle quali, corposissimo, si trova nella documentazione del pacchetto.

```
\si{\kilogram\metre\per\second}          \\
\si{\kilo\gram\metre\per\square\second} \\
\si{\gram\per\cubic\centi\metre}         \\
\si{\square\volt\cubic\lumen\per\farad} \\
\si{\metre\squared\per\gray\cubic\lux}  \\
\si{\henry\second}                       \\
\si[per-mode = symbol]%
  {\kilogram\metre\per\second}           \\
\si[per-mode = fraction]%
  {\kilogram\metre\per\ampere\per\second}
```

$\mathrm{kg\,m\,s^{-1}}$
$\mathrm{kg\,m\,s^{-2}}$
$\mathrm{g\,cm^{-3}}$
$\mathrm{V^{2}\,lm^{3}\,F^{-1}}$
$\mathrm{m^{2}\,Gy^{-1}\,lx^{3}}$
$\mathrm{H\,s}$
$\mathrm{kg\,m/s}$
$\mathrm{\frac{kg\,m}{A\,s}}$

È un metodo meno conveniente del primo, nonostante che si basi più sul significato che sull'aspetto, ma è utile per definire macro personalizzate, perché siunitx fornisce numerose abbreviazioni predefinite e nuove funzionalità. Inoltre, agendo sulle impostazioni del pacchetto, lo stesso testo sorgente può produrre risultati diversi: per esempio, la macro `\per` modificata con la chiave `per-mode` nel codice precedente può assumere a seconda dei casi il significato di potenza reciproca (impostazione predefinita), barra (valore `symbol`) o linea di frazione (valore `fraction`).

7 Quantità fisiche

Quantità fisiche pure

Per comporre le quantità fisiche c'è il comando

```
\SI[⟨opzioni⟩]{⟨numero⟩}[⟨preunità⟩]{⟨unità⟩}
```

che fonde le funzionalità di `\num` e `\si` accoppiando un ⟨*numero*⟩ a un'⟨*unità*⟩ e interponendo tra i due elementi lo spazio corretto. La ⟨*preunità*⟩, se presente, viene stampata *prima* del valore numerico (di solito è un simbolo di valuta).

```
\SI[mode = text]{1.2}{J.mol^{-1}.K^{-1}} \\
\SI{,23e7}{\candela}                     \\
\SI[per-mode = symbol]
  {1,99}[\$]{\per\kilogram}              \\
\SI[per-mode = fraction]
  {1,345}{\coulomb\per\mole}
```

$1{,}2\,\mathrm{J\,mol^{-1}\,K^{-1}}$
$0{,}23 \times 10^{7}\,\mathrm{cd}$
\$1,99/kg
$1{,}345\,\mathrm{\frac{C}{mol}}$

L'opzione `mode = text` nell'esempio precedente chiede a siunitx di comporre l'output con il font impostato per il testo. Si vedrebbe il risultato se in questo articolo si fossero scelti i numeri minuscoli: $1{,}23\,\mathrm{J\,mol^{-1}\,K^{-1}}$.

Liste di quantità fisiche

Per comporre liste di quantità fisiche c'è il comando

```
\SIlist[⟨opzioni⟩]{⟨numeri⟩}{⟨unità⟩}
```

che funziona come `\numlist`, ma aggiunge un'unità a ciascun valore numerico.

```
\SIlist{10;30;45}{\metre}
```

10 m, 30 m e 45 m

Intervalli di quantità fisiche

Per comporre intervalli di quantità fisiche c'è il comando

```
\SIrange[⟨opzioni⟩]{⟨numero1⟩}{⟨numero2⟩}%
  {⟨unità⟩}
```

che funziona come `\numrange`, ma aggiunge un'unità a ciascun numero.

```
\SIrange{0,13}{0,67}{\milli\metre}
```

0,13 mm a 0,67 mm

8 Macro

Altre macro predefinite

Oltre a quelle standard, siunitx definisce le seguenti macro particolari.

`\meter` | `\deka` Sostituiscono rispettivamente `\metre` e `\deca` per gli utenti americani.

`\celsius` Abbreviazione per `\degreeCelsius`.

`\percent` Non è un'unità, ma spesso è adoperata come tale.

`\square` | `\cubic` | `\squared` | `\cubed` Macro per potenze. Le prime due *precedono* l'unità; le altre la *seguono*.

`\tothe{`⟨*potenza*⟩`}` | `\raiseto{`⟨*potenza*⟩`}`
: Uniche tra le macro per le unità a prendere un argomento, servono a elevare *una tantum* una base a potenze generiche e precedono o seguono indifferentemente la base.

`\per` Serve per le potenze reciproche e per impostazione predefinita agisce solo sull'unità immediatamente successiva.

`\of` Serve a inserire qualificatori generici.

`\cancel` | `\highlight{`⟨*colore*⟩`}` Barrano e colorano le unità rispettivamente. Richiedono il pacchetto cancel.

`\si` Permette di stampare anche i soli prefissi delle unità, limitatamente a un prefisso per volta.

Ecco qualche esempio di quanto s'è appena descritto.

```
\si{\square\becquerel}                          \\
\si{\joule\squared\per\lumen}                  \\
\si{\cubic\lux\volt\tesla\cubed}               \\
\si{\henry\tothe{5}}                           \\
\si{\raiseto{4,5}\radian}                      \\
\si{\joule\per\mole\per\kelvin}                \\
\si{\joule\per\mole\kelvin}                    \\
\si{\per\henry\tothe{5}}                       \\
\si{\per\square\becquerel}                     \\
\si{\kilogram\of{metallo}}                     \\
\si{\kilo}                                     \\
\si{\micro}                                    \\
\si[prefixes-as-symbols = false]{\kilo}        \\
\si{\highlight{red}\kilogram\metre%
  \per\second}                                 \\
\si[unit-color = purple]{\highlight{red}%
  \kilogram\metre\per\second}                  \\
\si[per-mode = fraction]{\cancel\kilogram%
  \metre\per\cancel\kilogram\per\second}       \\
\SI[qualifier-mode = brackets]{0,1}{\milli%
  \mole\of{cat}\per\kilogram\of{prod}}
```

Bq^{2}
$\mathrm{J}^{2}\,\mathrm{lm}^{-1}$
$\mathrm{lx}^{3}\,\mathrm{V}\,\mathrm{T}^{3}$
H^{5}
$\mathrm{rad}^{4,5}$
$\mathrm{J\,mol^{-1}\,K^{-1}}$
$\mathrm{J\,mol^{-1}\,K}$
H^{-5}
Bq^{-2}
$\mathrm{kg_{metallo}}$
k
µ
10^{3}
$\mathrm{kg\,m\,s^{-1}}$
$\mathrm{kg\,m\,s^{-1}}$
$\frac{\cancel{\mathrm{kg}}\,\mathrm{m}}{\cancel{\mathrm{kg}}\,\mathrm{s}}$
$0{,}1\,\mathrm{mmol(cat)\,kg(prod)^{-1}}$

La chiave `prefixes-as-symbols` è un interruttore che permette di ottenere i prefissi delle unità come lettere o come potenze di numeri. Con la chiave `unit-color` si può specificare il colore scelto per l'unità. La chiave `qualifier-mode`, infine, si occupa dell'aspetto del qualificatore dell'unità.

8.1 Definire nuove macro

Tutte le macro descritte in questa sezione sono utilizzabili solo nel preambolo del documento.

Nuove unità di misura

Il comando

```
\DeclareSIUnit[⟨opzioni⟩]{⟨unità⟩}{⟨simbolo⟩}
```

serve a definire nuove unità di misura. Dichiarando

```
\DeclareSIUnit[number-unit-product = {}]%
  {\degree}{\SIUnitSymbolDegree}
```

si potrà scrivere[1]

```
\SI{3,1415}{\degree} \\
\SI[number-unit-product = \,]%
  {3,1415}{\degree}
```

3,1415°
3,1415 °

La chiave `number-unit-product` imposta il simbolo di prodotto tra numero e unità. Il comando `\SIUnitSymbolDegree` serve a rendere correttamente il simbolo del grado, problematico perché non letterale.

Nuovi prefissi e prefissi binari

I comandi

```
\DeclareSIPrefix{⟨prefisso⟩}{⟨simbolo⟩}%
  {⟨potenza-di-dieci⟩}
\DeclareBinaryPrefix{⟨prefisso⟩}{⟨simbolo⟩}%
  {⟨potenza-di-due⟩}
```

servono a definire nuovi prefissi e prefissi binari. Dichiarando

```
\DeclareSIPrefix{\kilo}{k}{3}
\DeclareBinaryPrefix{\kibi}{Ki}{10}
```

si potrà scrivere

```
\si{\kilo} \\
\si{\kibi}
```

k
Ki

Nuove potenze

I comandi

```
\DeclareSIPrePower{⟨potenza⟩}{⟨numero⟩}
\DeclareSIPostPower{⟨potenza⟩}{⟨numero⟩}
```

1. Le norme internazionali richiedono espressamente che i simboli come il `\degree`, che non contengono lettere, non siano staccati dal valore numerico della misura come invece è richiesto per le unità che contengono anche lettere. `\celsius`, perciò, è correttamente staccato; `\degree`, `\minute`, `\second`, invece, non dovrebbero *mai* essere staccati.

servono a definire nuove macro per le potenze, da anteporre o posporre all'unità di misura rispettivamente. Dichiarando

```
\DeclareSIPrePower{\quartica}{4}
\DeclareSIPostPower{\allaquarta}{4}
```

si potrà scrivere

```
\si{\kilogram%
  \allaquarta} \\
\si{\quartica\metre}
```

kg^{4}
m^{4}

Nuovi qualificatori

Il comando

```
\DeclareSIQualifier{⟨qualificatore⟩}%
  {⟨simbolo⟩}
```

serve a definire nuovi qualificatori. Dichiarando

```
\DeclareSIQualifier{\polimero}{pol}
\DeclareSIQualifier{\catalizzatore}{cat}
```

si potrà scrivere

```
\SI{1.234}{\gram\polimero\per\mole%
  \catalizzatore\per\hour}
```

$1{,}234\,\mathrm{g_{pol}\,mol_{cat}^{-1}\,h^{-1}}$

9 Tabelle: i fondamentali

In tutti i codici di qui in avanti si assume l'uso del pacchetto booktabs e per motivi di spazio si ometteranno gli elementi tipici degli ambienti galleggianti.

Colonne di soli numeri: `S`

Per colonne di numeri c'è lo specificatore `S`, che per impostazione predefinita mette il separatore decimale al centro della colonna e allinea opportunamente il resto del contenuto (tabella 2).

Tabella 2

```
\begin{tabular}{S}
  \toprule
    {Alcuni numeri} \\
  \midrule
       2,3456 \\
      34,2345 \\
      -6,7835 \\
      90,473  \\
    5642,5    \\
       1,2e3  \\
          e4  \\
  \bottomrule
\end{tabular}
```

I numeri vengono allineati correttamente anche in presenza di elementi non numerici prima o dopo. Nel caso in cui questi elementi potrebbero essere fraintesi da siunitx come numero o parte di numero, però, bisogna racchiuderli tra parentesi graffe, come si è fatto per l'intestazione della colonna.

Tabella 2: Comportamento predefinito di una colonna `S` di soli numeri.

Alcuni numeri
2,3456
34,2345
−6,7835
90,473
5642,1
$1{,}2 \times 10^{3}$
10^{4}

Tabella 3: Colorare i numeri: il comando `\color`.

Alcuni valori
12,34
975,31
44,268[a]

Colori

È possibile colorare il contenuto della tabella con il comando `\color` del pacchetto xcolor, il quale sovrascrive qualunque colore generale eventualmente applicato da siunitx (tabella 3).

Tabella 3

```
\begin{tabular}{S[color = orange]}
  \toprule
    {Alcuni valori} \\
  \midrule
    12,34                                  \\
    \color{purple} 975,31                  \\
    44,268 \textsuperscript{\emph{a}} \\
  \bottomrule
\end{tabular}
```

L'esponente della nota, però, *non* viene colorato.

Il comando `\tablenum`

Per allineare numeri nell'argomento dei comandi `\multicolumn` e `\multirow` c'è il comando

```
\tablenum[⟨opzioni⟩]{⟨numero⟩}
```

(tabella 4).

Tabella 4

```
\begin{tabular}{lr}
\toprule
  Intestazione & Intestazione \\
\midrule
  Informazioni & Ancora informazioni \\
  Informazioni & Ancora informazioni \\
  \multicolumn{2}{c}{
    \tablenum[table-format = 4.4]{12,34}
  } \\
```

Tabella 4: Allineamenti complessi: il comando `\tablenum` con `\multicolumn` (a sinistra) e con `\multirow` (a destra).

Intestazione	Intestazione
Informazioni	Ancora informazioni
Informazioni	Ancora informazioni
	12,34
	333,5567
	4563,21

Intestazione	Intestazione
88,999	aaa
	bbb
33,435	ccc
	ddd
	eee

Tabella 5: Colonna `s` di sole unità.

Unità
$\mathrm{m^2\,s^{-1}}$
Pa
$\mathrm{m\,s^{-1}}$

Tabella 6: Elaborazione degli elementi nelle colonne `s`.

Unità	Unità
$\mathrm{m^3}$	$\mathrm{m^3}$
kg	kg

```
  \multicolumn{2}{c}{
    \tablenum[table-format = 4.4]{333,5567}
  } \\
  \multicolumn{2}{c}{
    \tablenum[table-format = 4.4]{4563,21}
  } \\
\bottomrule
\end{tabular}
\hfil
\begin{tabular}{lr}
\toprule
  Intestazione & Intestazione \\
\midrule
  \multirow{2}*{\tablenum{88,999}} & aaa \\
                                   & bbb \\
  \multirow{2}*{\tablenum{33,435}} & ccc \\
                                   & ddd \\
                                   & eee \\
\bottomrule
\end{tabular}
```

L'opzione `table-format` è descritta più avanti.

Colonne di sole unità: `s`

Per colonne di sole unità c'è lo specificatore di colonna `s` (tabella 5).

Tabella5

```
\begin{tabular}{s}
  \toprule
    \multicolumn{1}{c}{Unità} \\
  \midrule
    \metre\squared\per\second \\
    \pascal                  \\
    m.s^{-1}                 \\
  \bottomrule
\end{tabular}
```

Siccome il comando `\si` accetta diversi tipi di input, la colonna `s` non è in grado di controllare il testo sorgente per vedere se è ben formato, perciò il suo *intero* contenuto viene passato a `\si` per essere elaborato. Per evitarlo, va adoperato il comando `\multicolumn` (tabella 6).

Tabella 6

```
\sisetup{color = orange}
\begin{tabular}{ss}
\toprule
  {Unità}    &
  \multicolumn{1}{c}{Unità}    \\
\midrule
  {\si{m^3}} &
  \multicolumn{1}{c}{\si{m^3}} \\
  \kilogram  & \kilogram       \\
\bottomrule
\end{tabular}
```

Si noti che né le parentesi da sole né la presenza di `\multicolumn` impediscono che il contenuto della cella venga passato a `\si` ed elaborato di conseguenza.

10 Opzioni specifiche per le tabelle

Oltre che dalle impostazioni descritte nelle sezioni precedenti, l'elaborazione del materiale tabellare è governata dalle opzioni specifiche descritte in questa sezione e raccolte nella tabella 7.

`table-parse-only`

Disattiva le funzionalità dello specificatore `S` nella colonna a cui viene applicata e mantiene attivo solo l'analizzatore numerico standard di siunitx (tabella 8).

Tabella 8

```
\begin{tabular}{
  S
  S[table-parse-only]
}
  \toprule
    {Centratura al separatore decimale} &
    {Centratura semplice}               \\
```

Tabella 7: Opzioni per le tabelle definite da siunitx ($\langle n \rangle$: intero).

Opzioni	Valori	Default
`table-align-comparator`	`true` | `false`	`true`
`table-align-exponent`	`true` | `false`	`true`
`table-align-text-pre`	`true` | `false`	`true`
`table-align-text-post`	`true` | `false`	`true`
`table-align-uncertainty`	`true` | `false`	`true`
`table-alignment`	`left` | `center` | `right`	
`table-auto-round`	`true` | `false`	`false`
`table-column-width`	Larghezza a piacere	`0pt`
`table-comparator`	`true` | `false`	`false`
`table-figures-decimal`	$\langle n \rangle$ in base all'input	`2`
`table-figures-exponent`	$\langle n \rangle$ in base all'input	`0`
`table-figures-integer`	$\langle n \rangle$ in base all'input	`3`
`table-figures-uncertainty`	$\langle n \rangle$ in base all'input	`0`
`table-format`	$\langle n \rangle.\langle n \rangle$ in base all'input	
`table-number-alignment`	`left` | `center` | `right` | `center-decimal-marker`	`center-decimal-marker`
`table-parse-only`	`true` | `false`	`false`
`table-omit-exponent`	`true` | `false`	`true`
`table-space-text-pre`	Testo a piacere	
`table-space-text-post`	Testo a piacere	
`table-sign-exponent`	`true` | `false`	`false`
`table-sign-mantissa`	`true` | `false`	`false`
`table-text-alignment`	`left` | `center` | `right`	`center`
`table-unit-alignment`	`left` | `center` | `right`	`center`

```
  \midrule
     12,345  &  12.345  \\
      6,78   &   6,78   \\
    -88,8(9) & -88,8(9) \\
      4,5e3  &   4,5e3  \\
\bottomrule
\end{tabular}
```

`table-number-alignment`

Controlla l'allineamento dei numeri rispetto ai margini della colonna `S`. Si noti che il valore predefinito funziona al meglio con un input più o meno simmetrico (stesso numero o quasi di cifre prima e dopo il separatore decimale) e che, se impostato, annulla molte delle altre opzioni eventualmente attive (tabella 9).

Tabella 9

```
\sisetup{
  table-figures-integer = 2,
  table-figures-decimal = 4
}
\begin{tabular}{
  S
  S[table-number-alignment = left]
  S[table-number-alignment = center]
  S[table-number-alignment = right]
}
  \toprule
   {Alcuni valori} & {Alcuni valori} &
   {Alcuni valori} & {Alcuni valori} \\
  \midrule
    2,3456 &  2,3456 &  2,3456 &  2,3456 \\
   34,2345 & 34,2345 & 34,2345 & 34,2345 \\
   56,7835 & 56,7835 & 56,7835 & 56,7835 \\
   90,473  & 90,473  & 90,473  & 90,473  \\
  \bottomrule
\end{tabular}
```

Le due opzioni `table-figures-...` sono descritte di seguito.

`table-figures-...`
`table-sign-...`

siunitx calcola lo spazio da riservare a un numero mediante due famiglie di opzioni. La prima, costituita dalle quattro opzioni seguenti, calcola quanto spazio assegnare alla parte del numero indicata dal nome dell'opzione dopo `figures-`.

`table-figures-decimal` Cifre decimali.

`table-figures-exponent` Esponenti.

`table-figures-integer` Cifre intere.

`table-figures-uncertainty` Incertezze.

Se il valore è impostato a `0`, non viene riservato alcuno spazio e alcuni elementi verranno messi scorrettamente sulla pagina o non stampati affatto (ma un `Warning` nel log segnalerà la cosa). Lo spazio riservato a una determinata parte di un numero comprenderà automaticamente anche quello necessario per qualunque altro oggetto a esso eventualmente associato (per esempio, il simbolo × per gli esponenti).

Tabella 8: Colonne `S` con i numeri allineati (a sinistra) e non allineati: (a destra): opzione `table-parse-only`.

Centratura al separatore decimale	Centratura semplice
12,345	12,345
6,78	6,78
−88,8(9)	−88,8(9)
$4{,}5 \times 10^3$	$4{,}5 \times 10^3$

Tabella 9: Diversi allineamenti: opzione `table-number-alignment`.

Alcuni valori	Alcuni valori	Alcuni valori	Alcuni valori
2,3456	2,3456	2,3456	2,3456
34,2345	34,2345	34,2345	34,2345
56,7835	56,7835	56,7835	56,7835
90,473	90,473	90,473	90,473

Le due opzioni della seconda famiglia sono interruttori che determinano se lo spazio per un segno è stato effettivamente riservato o no.

`table-sign-exponent` Negli esponenti.

`table-sign-mantissa` Nella parte decimale.

Il comportamento di alcune di queste impostazioni è mostrato nella tabella 10.

Tabella 10

```
\sisetup{
  table-number-alignment = center,
  table-figures-integer  = 2
}
\begin{tabular}{
  S
  S[table-number-alignment = right]
  S[table-figures-uncertainty = 1]
  S[
    separate-uncertainty,
    table-figures-uncertainty = 1
  ]
  S[table-sign-mantissa]
  S[table-figures-exponent = 1]
}
  \toprule
    {Valori} & {Valori} & {Valori} &
    {Valori} & {Valori} & {Valori} \\
  \midrule
     2,3     &   2,3  &  2,3(5)  &
     2,3(5)  &   2,3  &  2,3e8   \\
    34,23    &  34,23 & 34,23(4) &
    34,23(4) &  34,23 & 34,23    \\
    56,78    &  56,78 & 56,78(3) &
    56,78(3) & -56,78 & 56,78e3  \\
     3,76    &   3,76 &  3,76(2) &
     3,76(2) & +-3,76 &    e6    \\
  \bottomrule
\end{tabular}
```

L'opzione `separate-uncertainty` separa l'incertezza dal numero che la contiene.

`table-comparator`

Calcola lo spazio per i comparatori (tabella 11).

Tabella 11

```
\sisetup{
  table-number-alignment = center,
  table-figures-integer  = 2,
  table-figures-decimal  = 2,
  table-figures-exponent = 2
}
\begin{tabular}{
  S
  S[table-comparator = true]
}
  \toprule
    {Valori} & {Valori} \\
  \midrule
     2,3     & <   2,3e8  \\
    34,23    & =  34,23   \\
    56,78    & >= 56,78e3 \\
     3,76    & \gg     e6 \\
  \bottomrule
\end{tabular}
```

Le macro interne che elaborano i dati ometteranno tutte le parti di un numero senza spazio riservato, notificandolo con un `Warning`. Ciò significa che se non si riserva loro dello spazio, incertezze ed esponenti non verranno stampati.

`table-format`

È una comoda scorciatoia per le sei opzioni `table-figures-...` e `table-sign-...` già viste, in quanto permette di passare le stesse informazioni a siunitx in modo 'compresso'. Il suo valore è un'espressione del tipo $\langle n \rangle . \langle n \rangle$, dove $\langle n \rangle$ indica il numero di cifre in ciascuna parte del numero. Così,

```
\sisetup{table-format = 3.2}
```

equivale a

Tabella 10: Riservare spazio a numeri e segni: opzioni `table-figures-...` e `table-sign-...`

Valori	Valori	Valori	Valori	Valori	Valori
2,3	2,3	2,3(5)	$2{,}3 \pm 0{,}5$	2,3	$2{,}3 \times 10^{8}$
34,23	34,23	34,23(4)	$34{,}23 \pm 0{,}04$	34,23	34,23
56,78	56,78	56,78(3)	$56{,}78 \pm 0{,}03$	−56,78	$56{,}78 \times 10^{3}$
3,76	3,76	3,76(2)	$3{,}76 \pm 0{,}02$	±3,76	10^{6}

Tabella 11: Riservare spazio ai comparatori: opzione `table-comparator`.

Valori	Valori
2,3	$< 2{,}3 \times 10^{8}$
34,23	$=34{,}23$
56,78	$\geq 56{,}78 \times 10^{3}$
3,76	$\gg 10^{6}$

```
\sisetup{
  table-figures-integer = 3,
  table-figures-decimal = 2
}
```

Sarà interpretata correttamente anche la presenza di un segno, così

```
\sisetup{table-format = +3.2e+4}
```

avrà lo stesso effetto di

```
\sisetup{
  table-figures-integer  = 3,
  table-figures-decimal  = 2,
  table-figures-exponent = 4,
  table-sign-mantissa,
  table-sign-exponent
}
```

È importante notare che per tutte le parti di un numero *non* specificate nell'argomento di `table-format` il numero di cifre è impostato a zero e che, adoperando questa opzione, `table-number-alignment` viene portato a `center` (tabella 13).

Tabella 13

```
\begin{tabular}{
  S
  S[table-format =  2.2]
  S[table-format =  2.2(1)]
  S[table-format = +2.2]
  S[table-format =  2.2e1]
  }
  \toprule
    {Valori} & {Valori} & {Valori} &
    {Valori} & {Valori} \\
  \midrule
      2,3  &  2,3  &  2,3(5)  &
      2,3  &  2,3e8  \\
     34,23 & 34,23 & 34,23(4) &
     34,23 & 34,23   \\
     56,78 & 56,78 & 56,78(3) &
    -56,78 & 56,78e3 \\
      3,76 &  3,76 &  3,76(2) &
    +-3,76 &    e6   \\
  \bottomrule
\end{tabular}
```

Tabella 12: Testo prima e dopo i numeri: opzioni `table-space-text-pre` e `...-post`.

Valori
2,3456
34,2345[a]
56,7835
ora 90,473

table-space-text-pre
table-space-text-post

Prendono come valore dell'eventuale materiale aggiuntivo da mettere prima o dopo un numero rispettivamente. siunitx calcolerà lo spazio da riservare al testo inserito mantenendo l'allineamento dei numeri (tabella 12).

Tabella 12

```
\sisetup{
  table-number-alignment = center,
  table-figures-integer  = 2,
  table-figures-decimal  = 4,
  table-space-text-pre   = ora~,
  table-space-text-post  =
    \textsuperscript{\emph{a}}
}
\begin{tabular}{S}
  \toprule
    {Valori} \\
  \midrule
     2,3456                              \\
    34,2345 \textsuperscript{\emph{a}} \\
    56,7835                              \\
    ora~90,473                           \\
  \bottomrule
\end{tabular}
```

table-align-exponent
table-align-uncertainty
table-align-comparator

La prima opzione permette di allineare gli esponenti o di avvicinarli alla parte decimale (tabella 14 a sinistra).

Tabella 13: Un modo alternativo per riservare spazio a numeri e segni: opzione `table-format`.

Valori	Valori	Valori	Valori	Valori
2,3	2,3	2,3(5)	2,3	$2,3 \times 10^{8}$
34,23	34,23	34,23(4)	34,23	34,23
56,78	56,78	56,78(3)	−56,78	$56,78 \times 10^{3}$
3,76	3,76	3,76(2)	±3,76	10^{6}

Tabella 14 a sinistra

```
\sisetup{
  table-format           = 1.3e2,
  table-number-alignment = center
}
\begin{tabular}{
  S
  S[table-align-exponent = false]
}
  \toprule
    {Intestazione} & {Intestazione} \\
  \midrule
    1,2e3          & 1,2e3    \\
    1,234e56       & 1,234e56 \\
  \bottomrule
\end{tabular}
```

La seconda, permette di allineare le incertezze separandole dalla parte decimale o di avvicinarle a essa (tabella 14 al centro).

Tabella 14 al centro

```
\sisetup{
  separate-uncertainty,
  table-format = 1.3(1)
}
\begin{tabular}{
  S
  S[table-align-uncertainty = false]
}
  \toprule
    {Intestazione} & {Intestazione} \\
  \midrule
    1,2(1)         & 1,2(3)   \\
    1,234(5)       & 1,234(5) \\
  \bottomrule
\end{tabular}
```

La terza, infine, permette di fare la stessa cosa con i comparatori (tabella 14 a destra).

Tabella 14 a destra

```
\sisetup{table-format = >2.2}
\begin{tabular}{
  S
  S[table-align-comparator = false]
}
  \toprule
    {Intestazione} & {Intestazione} \\
  \midrule
    > 1,2          & > 1,2  \\
    <12,34         & <12,34 \\
  \bottomrule
\end{tabular}
```

table-omit-exponent

Omette gli esponenti nel corpo della tabella per renderla più chiara nei casi in cui i dati da ordinare coprano un intervallo di valori e si renda necessario mettere un esponente fisso nell'intestazione (tabella 15).

Tabella 15

```
\begin{tabular}{
  S[table-format = 1.1e1]
  S[
    fixed-exponent = 3,
    table-format   = 2.1,
    table-omit-exponent
  ]
}
  \toprule
    {Intestazione} &
    {Intestazione\,/\,\num{e3}} \\
  \midrule
    1.2e3          & 1,2e3 \\
      3e2          &   3e2 \\
    1.0e4          & 1,0e4 \\
  \bottomrule
\end{tabular}
```

L'opzione imposta automaticamente per la colonna `scientific-notation = fixed`, che converte i numeri in notazione scientifica secondo il valore dell'esponente assegnato a `fixed-exponent`.

table-align-text-pre
table-align-text-post

Permettono di avvicinare al numero il contrassegno di una nota, che nelle tabelle è indicato spesso *dopo* il numero, a seconda che si trovi prima o dopo il numero rispettivamente (tabella 16).

Tabella 16

```
\newrobustcmd\NoteMark[1]{%
  \textsuperscript{\emph{#1}}%
}
\sisetup{
  table-number-alignment = center,
  table-figures-integer  = 2,
  table-figures-decimal  = 4,
  table-space-text-pre   = \NoteMark{a}
}
```

Tabella 14: Allineare esponenti (opzione `table-align-exponent`, a sinistra); incertezze (opzione `table-align-uncertainty`, al centro); comparatori (opzione `table-align-comparator`, a destra).

Intestazione	Intestazione
$1{,}2 \times 10^{3}$	$1{,}2 \times 10^{3}$
$1{,}234 \times 10^{56}$	$1{,}234 \times 10^{56}$

Intestazione	Intestazione
$1{,}2 \pm 0{,}1$	$1{,}2 \pm 0{,}3$
$1{,}234 \pm 0{,}005$	$1{,}234 \pm 0{,}005$

Intestazione	Intestazione
$> 1{,}2$	$>1{,}2$
$<12{,}34$	$<12{,}34$

Tabella 15: Eliminare gli esponenti dal corpo della tabella: opzione `table-omit-exponent`.

Intestazione	Intestazione/10^3
$1{,}2 \times 10^{3}$	1,2
3×10^{2}	0,3
$1{,}0 \times 10^{4}$	10

```
\begin{tabular}{
  S
  S[table-align-text-pre = false]
}
  \toprule
    {Valori}             & {Valori} \\
  \midrule
    2,3456               & 2,3456   \\
    \NoteMark{a}  4,234  &
    \NoteMark{a}  4,234  \\
    \NoteMark{b}   ,78   &
    \NoteMark{b}   ,78   \\
    \NoteMark{d} 88      &
    \NoteMark{d} 88      \\
\bottomrule
\end{tabular}
\hfil
\sisetup{
  table-space-text-post = \NoteMark{a}
}
\begin{tabular}{
  S
  S[table-align-text-post = false]
}
  \toprule
    {Valori}            & {Valori} \\
  \midrule
     2,3456             &  2,3456  \\
    34,234 \NoteMark{a} &
    34,234 \NoteMark{a} \\
    56,78  \NoteMark{b} &
    56,78  \NoteMark{b} \\
    90,4   \NoteMark{c} &
    90,4   \NoteMark{c} \\
    88     \NoteMark{d} &
    88     \NoteMark{d} \\
  \bottomrule
\end{tabular}
```

table-auto-round

Arrotonda o riempie di zeri automaticamente il contenuto di una colonna al numero di cifre decimali indicato come valore di `table-figures-decimal` (tabella 17).

Tabella 16: Avvicinare i contrassegni delle note ai numeri: opzioni `table-align-text-pre` (a sinistra) e `...-post` (a destra).

Valori	Valori
2,3456	2,3456
[a] 4,234	[a]4,234
[b] 0,78	[b]0,78
[c]25,3	[c]12,7
[d]88	[d]88

Valori	Valori
2,3456	2,3456
34,234 [a]	34,234[a]
56,78 [b]	56,78[b]
90,4 [c]	90,4[c]
88 [d]	88[d]

Tabella 17: Arrotondamenti automatici: opzione `table-auto-round`.

Intestazione	Intestazione
1,2	1,200
1,2345	1,235

Tabella 17

```
\sisetup{
  table-number-alignment = center,
  table-figures-integer  = 1,
  table-figures-decimal  = 3
}
\begin{tabular}{
  S
  S[table-auto-round]
}
  \toprule
    {Intestazione} & {Intestazione} \\
  \midrule
    1,2            & 1,2            \\
    1,2345         & 1,2345         \\
  \bottomrule
\end{tabular}
```

parse-numbers

Attiva e disattiva l'analizzatore numerico di siunitx. È utile impostarla su `false` quando tutti i numeri in un documento debbono essere stampati 'come scritti', alleggerendo così di molto i calcoli che TEX deve eseguire per elaborare i dati. Richiedendola in una tabella, il contenuto della colonna verrà allineato al primo separatore decimale e stampato in modo matematico come sempre, ma questa volta lo spazio per i numeri sarà calcolato tenendo conto solo dei valori interi e decimali per la parte decimale (tabella 18).

Tabella 18: Allineamento senza analisi numerica (seconda, terza e quarta colonna): opzione `parse-numbers`.

Alcuni valori	Alcuni valori	Alcuni valori	Alcuni valori
2,35	2,35	2,35	2,35
34,234	34,234	34,234	34,234
56,783	56,783	56,783	56,783
3,762	3,762	3,762	3,762
$\sqrt{2}$	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{2}$

Tabella 18

```
\sisetup{
  parse-numbers = false,
  table-figures-integer = 2,
  table-figures-decimal = 3
}
\begin{tabular}{
  S
  S[table-number-alignment = center]
  S[table-number-alignment = right]
  S[table-number-alignment = left]
}
  \toprule
    {Alcuni valori} & {Alcuni valori} &
    {Alcuni valori} & {Alcuni valori} \\
  \midrule
     2,35  &  2,35  &  2,35  &  2,35  \\
    34,234 & 34,234 & 34,234 & 34,234 \\
    56,783 & 56,783 & 56,783 & 56,783 \\
     3,762 &  3,762 &  3,762 &  3,762 \\
    \sqrt{2} & \sqrt{2} &
    \sqrt{2} & \sqrt{2} \\
  \bottomrule
\end{tabular}
```

Si può cogliere la differenza osservando il diverso allineamento della radice nella prima colonna e in quelle successive.

table-text-alignment

Allinea il contenuto delle celle che non contengono alcun dato numerico (tabella 19).

Tabella 19

```
\sisetup{
  table-number-alignment = center,
  table-figures-integer  = 4,
  table-figures-decimal  = 4
}
\begin{tabular}{
  S[table-text-alignment = left]
  S
  S[table-text-alignment = right]
}
  \toprule
    {Valori}  & {Valori}  & {Valori}  \\
  \midrule
     992,435  &  992,435  &  992,435  \\
    7734,2344 & 7734,2344 & 7734,2344 \\
      56,7834 &   56,7834 &   56,7834 \\
       3,7462 &    3,7462 &    3,7462 \\
  \bottomrule
\end{tabular}
```

Tabella 19: Allineare contenuti non numerici: opzione `table-text-alignment`.

Valori	Valori	Valori
992,435	992,435	992,435
7734,2344	7734,2344	7734,2344
56,7834	56,7834	56,7834
3,7462	3,7462	3,7462

Tabella 20: Allineare le unità: opzione `table-unit-alignment`.

A sinistra	Centrato	A destra
$\mathrm{m\,s^{-1}}$	$\mathrm{m\,s^{-1}}$	$\mathrm{m\,s^{-1}}$
kg	kg	kg

table-unit-alignment

Allinea il contenuto delle colonne `s` (tabella 20).

Tabella 20

```
\begin{tabular}{
  s[table-unit-alignment = left]
  s
  s[table-unit-alignment = right]
}
  \toprule
    {A sinistra}   &
    {Centrato}       &
    {A destra} \\
  \midrule
    \metre\per\second &
    \metre\per\second &
    \metre\per\second \\
    \kilogram         &
    \kilogram         &
    \kilogram         \\
  \bottomrule
\end{tabular}
```

table-alignment

Permette di impostare al medesimo valore in una volta sola tutte e tre le opzioni di allineamento viste più sopra (`table-...-alignment`).

Tabella 21: Colonne di larghezza fissa: opzione `table-column-width`.

Flessibile	Fissa	Flessibile	Fissa
$\mathrm{m\,s^{-1}}$	$\mathrm{m\,s^{-1}}$	1,23	1,23
kg cd	kg cd	45,6	45,6

table-column-width

Permette di impostare la larghezza di una colonna a un valore fisso nei casi in cui il comportamento predefinito non sia quello desiderato (tabella 21).

Tabella 21

```
\begin{tabular}{
  s
  s[table-column-width = 2 cm]
  S
  S[table-column-width = 2 cm]
}
  \toprule
    {Flessibile} & {Fissa} &
    {Flessibile} & {Fissa} \\
  \midrule
    \metre\per\second &
    \metre\per\second &
    1,23 &  1,23 \\
    \kilogram\candela &
    \kilogram\candela &
    45,6 & 45,6  \\
  \bottomrule
\end{tabular}
```

La si può adoperare anche per ottenere effetti particolari: per esempio, per centrare globalmente una colonna di numeri sotto un'intestazione lunga, con i numeri allineati a propria volta a destra (tabella 22).

Tabella 22

```
\settowidth\mylength{Intestazione lunga}
\sisetup{
  table-format           = 4,
  table-number-alignment = center,
  table-column-width     = \mylength,
  input-decimal-markers  = ,
  input-symbols          = .
}
\begin{tabular}{S}
  \toprule
    {Intestazione lunga} \\
  \midrule
      1,33 \\
       2   \\
    1234   \\
  \bottomrule
\end{tabular}
```

La chiave `input-decimal-markers` permette di specificare il carattere da considerare come separatore decimale nell'input (nessuno, in questo caso); la chiave `input-symbols` qui indica che il punto è legale in una colonna `S` (ma non sarà considerato un separatore per via dell'opzione precedente).

Tabella 22: Colonna centrata sotto un'intestazione lunga con numeri allineati a destra: opzione `table-column-width`.

Intestazione lunga
12,33
2
1234

11 Usi avanzati di siunitx

11.1 Aggiungere elementi dopo l'ultima colonna di una tabella

Adoperando il costrutto '`<`' del pacchetto array per inserire materiale dopo una colonna `S` o `s`, l'allineamento dell'ultima colonna della tabella potrebbe risultare sbagliato se si terminano le righe con il comando standard `\\`. Si può risolvere il problema sostituendolo con la primitiva `\cr` (tabella 23).

Tabella 23

```
\hfil
\begin{tabular}{
  S<{\,\si{\kg}}
  S<{\,\si{\kg}}
}
 \toprule
  \multicolumn{1}{c}{Intestazione lunga} &
  \multicolumn{1}{c}{Intestazione lunga} \\
 \midrule
   1,23 & 1,23 \\
   4,56 & 4,56 \\
   7,8  & 7,8  \\
 \bottomrule
\end{tabular}
\hfil
\begin{tabular}{
  S<{\,\si{\kg}}
  S<{\,\si{\kg}}
}
 \toprule
  \multicolumn{1}{c}{Intestazione lunga} &
  \multicolumn{1}{c}{Intestazione lunga} \\
 \midrule
   1,23 & 1,23 \cr
   4,56 & 4,56 \cr
   7,8  & 7,8  \cr
 \bottomrule
\end{tabular}
\hfil
```

Tabella 23: Allineamento con array in una colonna S a fine tabella (scorretto, a sinistra; corretto, a destra): la primitiva \cr.

Intestazione lunga	Intestazione lunga
1,23 kg	1,23 kg
4,56 kg	4,56 kg
7,8 kg	7,8 kg

Intestazione lunga	Intestazione lunga
1,23 kg	1,23 kg
4,56 kg	4,56 kg
7,8 kg	7,8 kg

Tabella 24: Incolonnare numeri irrelati.

Intestazione
120
12,3
12340
12,02
123
1

11.2 Colonne di numeri irrelati

Quando si ha una colonna di numeri irrelati, il solito consiglio è di allinearli a destra e centrare la colonna risultante sotto l'intestazione. Questa sorta di 'abuso' della natura di un numero si può perpetrare anche con siunitx (tabella 24).

Tabella 24

```
\begin{tabular}{
  S[
    table-format           = 5.0,
    parse-numbers          = false,
    input-symbols          = ,,
    input-decimal-markers = x
  ]
}
  \toprule
    \multicolumn{1}{c}{Intestazione} \\
  \midrule
    120   \\
    12,3  \\
    12340 \\
    12,02 \\
    123   \\
    1     \\
  \bottomrule
\end{tabular}
```

11.3 Colonne di quantità fisiche irrelate

Di solito, in una tabella le unità andrebbero scritte nell'intestazione della colonna e non accanto a ciascun numero, ma in alcuni casi le quantità fisiche da sistemare in una tabella sono disomogenee. Ci sono due metodi per risolvere il problema.

Il primo consiste nel mettere le unità nella prima colonna, il che ha senso se la tabella contiene più elementi correlati a ciascuna unità.

Il secondo metodo consiste nel generare due colonne, una per i numeri e una per le unità, e poi formattarle in modo da simulare alla vista l'effetto di un'unica colonna. Il trucco è adatto per tabelle che presentano una sola serie di quantità fisiche.

Tabella 25: Tabelle di quantità fisiche disomogenee.

	Uno	Due	Intestazione
a/Å	1,234(2)	5,678(4)	1,234 m
β/°	90,34(4)	104,45(5)	0,835 cd
μ/mm^{-1}	0,532	0,894	4,23 J mol^{-1}

La tabella 25 mostra quanto si è appena descritto (richiede il pacchetto array).

Tabella 25

```
\hfil
\begin{tabular}{
  >{$}l<{$}
  S[table-format = 2.3(1)]
  S[table-format = 3.3(1)]
}
  \toprule
    & {Uno} & {Due} \\
  \midrule
    a/\si{\angstrom}   &
    1,234(2) &   5,678(4) \\
    \beta/\si{\degree} &
    90,34(4) & 104,45(5)  \\
    \mu/\si{\per\mm}   &
    0,532    &   0,894    \\
  \bottomrule
\end{tabular}
\hfil
\begin{tabular}{
  S[table-format = 1.3]@{\,}
  s[table-unit-alignment = left]
}
  \toprule
    \multicolumn{2}{c}{Intestazione} \\
  \midrule
    1,234 & \metre          \\
    0,835 & \candela        \\
    4,23  & \joule\per\mole \\
  \bottomrule
\end{tabular}
\hfil
```

11.4 Tabelle con intestazioni

Una formattazione comune per le tabelle consiste nel rendere visivamente distinta la riga dell'intestazione con il testo in nero e un colore di sfondo. Se l'intestazione contiene numeri in una colonna S, però, ottenere l'effetto desiderato potrebbe essere

Tabella 26: Tabella con riga d'intestazione.

123,456
23,45
123,4
3,456

difficile. L'approccio migliore consiste nel rendere la macro `\bfseries` robusta, quindi adoperarla per produrre il testo dell'intestazione in nero (tabella 26).

Tabella 26

```
\robustify\bfseries
\begin{tabular}{
  S[
    detect-weight,
    table-format = 3.3
  ]
}
  \rowcolor[gray]{0.9}
  \bfseries 123,456 \\
             23,45  \\
            123,4   \\
              3,456 \\
\end{tabular}
```

11.5 Uso delle unità in tabelle e grafici

Ripetere le unità dopo ogni elemento in una tabella o sotto ogni marcatore degli assi di un grafico genera confusione ed è ripetitivo. Altrettanto comunemente si mettono le unità tra parentesi quadre, ma è un'operazione matematicamente povera. Molto meglio, invece, è separare tutti i valori numerici dall'unità, mettendoli così come sono nella colonna. La tabella 27 e la figura 1 mostrano quanto appena suggerito.

Tabella 27

```
\sisetup{
  table-number-alignment = center,
  table-figures-integer  = 1,
  table-figures-decimal  = 4
}
\begin{tabular}{@{}cS@{}}
  \toprule
    Elemento & {Lunghezza/\si{\metre}} \\
  \midrule
    1        & 1,1234 \\
    2        & 1,1425 \\
    3        & 1,7578 \\
    4        & 1,9560 \\
  \bottomrule
\end{tabular}
```

Figura 1

```
\begin{tikzpicture}
\begin{axis}[
  xmax = 6, xmin = 0, ymin = 0,
  width = 7cm, height = 6cm,
```

Tabella 27: Corretta intestazione di una tabella.

Elemento	Lunghezza/m
1	1,1234
2	1,1425
3	1,7578
4	1,9560

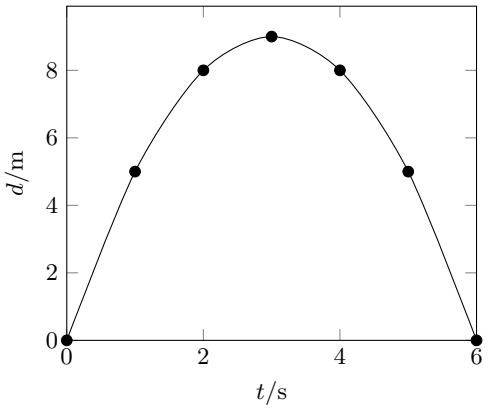

Figura 1: Corretta etichettatura di un grafico.

```
  xlabel = $t/\si{\second}$,
  ylabel = $d/\si{\metre}$,
]
  \addplot[smooth, mark = *]
    plot coordinates {
      (0,0) (1,5)
      (2,8) (3,9)
      (4,8) (5,5)
      (6,0)
  };
\end{axis}
\end{tikzpicture}
```

Infine, in molti casi è meglio mettere un esponente fisso nell'intestazione anziché scriverlo accanto a ciascun numero nel corpo della tabella, come mostra la tabella 28.

Tabella 28

```
\sisetup{table-number-alignment = center}
\begin{tabular}{
  c
  S[
    table-figures-integer  = 1,
    table-figures-decimal  = 3,
    table-figures-exponent = 1
  ]
  @{\,\si{\kilogram}}
  S[
    table-figures-integer = 2,
    table-figures-decimal = 2
  ]
}
  \toprule
    Elemento & \multicolumn{1}{c}{Massa} &
    {Massa/\SI{e3}{\kilogram}} \\
  \midrule
```

Tabella 28: Tabella con esponenti: ripetuti (al centro) e nell'intestazione (a destra).

Elemento	Massa	Massa/10^3 kg
1	4,56 $\times 10^3$ kg	4,56
2	2,40 $\times 10^3$ kg	2,40
3	1,345 $\times 10^4$ kg	13,45
4	4,5 $\times 10^2$ kg	0,45

```
    1 & 4,56e3  &  4,56 \\
    2 & 2,40e3  &  2,40 \\
    3 & 1,345e4 & 13,45 \\
    4 & 4,5e2   &  0,45 \\
  \bottomrule
\end{tabular}
```

Riferimenti bibliografici

WRIGHT, Joseph (2018). *siunitx — A comprehensive (SI) units package*. http://ctan.mirror.garr.it/mirrors/CTAN/macros/latex/contrib/siunitx/siunitx.pdf. v2.8b 2020/02/25.

▷ Joseph Wright
University of East Anglia
Norwich UK
joseph dot wright at
morningstar2 dot co dot uk

Ti*k*Z vs curve2e
Confronto sul disegno di un logo

Claudio Beccari, Gianluca Pignalberi

Sommario

Quale pacchetto conviene usare tra curve2e e Ti*k*Z quando il compito è disegnare un semplice logo alfabetico?

Abstract

Which package is more convenient to use when the task is to draw a simple, alphabetic logo: curve2e or Ti*k*Z?

La sfida

L'intera tiratura di ArsTEXnica del 2019, consegnata in occasione del GUIT*meeting2019*, ci è stata regalata da una casa editrice universitaria torinese: la CLUT — Cooperativa Libraria Universitaria di Torino (www.clut.it). Tale casa editrice, avente sede all'interno del Politecnico di Torino, è nota tra gli addetti ai lavori per essere una delle poche italiane a preferire la ricezione dei manoscritti in LaTeX. La sua classe di composizione è opera di Claudio Beccari.

Naturalmente avevamo bisogno di un loro logo da stampare sulla rivista. Ne abbiamo avuti due: uno vettoriale coi soli contorni delle lettere (opera di Claudio Beccari) e uno raster con le lettere colorate, ma troppo piccolo. Visto lo scarso tempo per andare in stampa, la redazione di ArsTEXnica ha realizzato una versione vettoriale, imprecisa ma passabile, con Inkscape: ogni lettera è stata disegnata unendo una serie di rettangoli e semicerchi.

Quest'anno la CLUT ha affidato a Gianluca Pignalberi e a Massimiliano Dominici la composizione del libro Barbara Bonelli, *Appunti di chimica — Terza edizione* e, a tal proposito, i compositori hanno deciso di realizzare un logo "filologicamente corretto" con l'aiuto di uno dei pacchetti di LaTeX che lo permettono. Ogni lettera del logo è contenuta in un quadrato di lato 3,4 cm e ognuna è distanziata dall'altra di 5 mm. La scelta è caduta su Ti*k*Z (Tantau, 2020).

Questo è il codice (le righe di ogni tracciato sono state qui suddivise su più righe solo per motivi di spazio):

```
\begin{tikzpicture}
\draw [fill] (1.7,0)%
arc [start angle=270, end angle=90,%
radius=1.7] -- ++(1.7,0)%
-- ++(0,-1.5) -- ++(-1.7,0)%
 arc [start angle=90, end angle=270,%
 radius=.2] -- ++(1.7,0)%
-- ++(0,-1.5) -- cycle; % C
\draw [fill,xshift=3.9cm] (0,0)%
-- ++(0,3.4) -- ++(1.5,0)%
-- ++ (0,-1.9) -- ++(1.9,0)%
-- ++(0,-1.5) -- cycle; % L
\draw [fill,xshift=7.8cm] (0,1.7)%
-- ++(0,1.7) -- ++(1.5,0)%
-- ++(0,-1.7)%
 arc [start angle=180, end angle=360,%
 radius=.2] -- ++(0,1.7) -- ++ (1.5,0)%
-- ++(0,-1.7) arc [start angle=0,%
 end angle=-180,radius=1.7]; % U
 \draw [fill,xshift=11.7cm] (.95,0)%
-- ++(0,1.9) -- ++(-.95,0)%
-- ++(0,1.5) -- ++(3.4,0)%
-- ++(0,-1.5) -- ++(-.95,0)%
-- ++(0,-1.9) -- cycle; % T
\end{tikzpicture}
```

Il metodo usato è banale: ogni lettera è disegnata in un quadrato con lato di 3,4 cm e con l'angolo in basso a sinistra posto nell'origine e poi traslata orizzontalmente alla distanza giusta. Il disegno è un banale inseguimento in senso orario del contorno con la prima coordinata assoluta (posta il più possibile vicino all'origine) e tutte le altre relative — lo stesso metodo del linguaggio Logo (Logo Foundation, 2020) — e il poligono mistilineo così creato viene poi riempito di nero.

Informato dell'intenzione di creare il logo con Ti*k*Z, Claudio Beccari ne ha realizzata una versione disegnata con curve2e, (Beccari, 2020), il cui codice è il seguente (anche in questo caso le righe troppo lunghe sono suddivise su più righe per la stampa):

```
\newcommand\CLUTlogo[2][nero]{%
\bgroup
\def\tempA{#2}\def\tempB{nero}%
\def\tempC{#1}%
\setbox0\hbox{C}\edef\CLUTsize{%
   \ifx\tempA\empty\ht0\else#2\fi}%
\unitlength=\dimexpr\CLUTsize/34\relax
\picture(154,34)
\unless\ifx\tempB\tempC\color{#1}\fi
\buttcap \linethickness{15\unitlength}
\Curve(34,27.5)<-1,0>(17.5,27.5)<-1,0>%
   (17.5,7.5)<1,0>(34,7.5)<1,0>
\put(40,0){\polyline[\miterjoin](7.5,34)%
```

Tabella 1: Confronto di quattro proprietà dei documenti prodotti.

Misura	TikZ (contorno)	TikZ ("scrittura")	curve2e
Lunghezza[a] (byte)	785	477	699
Numero di righe[b]	4	4	7
Dimensione PDF (byte)	1323	1225	1086
Compilazione (s)	0,369	0,229	0,196
Compilazione[c] (s)	0,204	0,200	0,180

[a] Dell'intero documento standalone.
[b] Del solo codice negli ambienti (tikz)picture, escluse le definizioni di comando e di apertura e chiusura ambiente.
[c] Esclusa la prima compilazione.

```
   (7.5,7.5)(34,7.5)}
\put(80,0){%
  \Curve(7.5,34)<0,-1>(7.5,17.5)<0,-1>%
    (27.5,17.5)<0,1>(27.5,34)<0,1>}
\put(120,0){\segment(0,27.5)(34,27.5)
  \segment(17,0)(17,34)}
\endpicture
\egroup}
```

Il codice realizzato consente di specificare la grandezza e il colore del logo; per ottenere il logo delle stesse dimensioni di quello ottenuto con TikZ, basta specificare `\CLUTlogo{34mm}`; se non si specifica nessuna dimensione (`\CLUTlogo{}`), il logo ha la stessa altezza delle maiuscole nel font corrente. A parte questa variante, il codice vero e proprio per disegnare il logo è compreso fra `\picture` e `\endpicture`. Esso si basa sul tracciamento di spezzate mistilinee di spessore pari a 15 unità di disegno.

Ispirato dalla soluzione proposta da Beccari e come suggerito da un anomino *referee*, Pignalberi ha realizzato una seconda versione TikZ in cui i tracciati sono disegnati esattamente come si scriverebbero le singole lettere. Anche l'ordine e la direzione ricalcano le sue mosse quando scrive. Ecco il nuovo codice:

```
\begin{tikzpicture} [line width=1.5cm]
\draw (3.4,2.65) -- ++(-1.7,0)%
arc [start angle=90,end angle=270,%
radius=.95] -- ++(1.7,0); % C
\draw [xshift=3.9cm] (.75,3.4)%
-- ++(0,-2.65) -- ++(2.65,0); % L
\draw [xshift=7.8cm] (.75,3.4)%
-- ++(0,-1.7) arc [start angle=180,%
end angle=360,radius=.95]%
-- ++(0,1.7); % U
\draw [xshift=11.7cm] (1.7,3.4)%
-- ++(0,-3.4) (0,2.65) -- ++(3.4,0); % T
\end{tikzpicture}
```

Il risultato dei tre disegni, visualmente identico, è mostrato nella figura 1.

A un'analisi "visuale", la prima versione TikZ è molto prolissa; il codice di Beccari è senz'altro più compatto e ancor più stringata è la seconda versione TikZ. La sintassi di curve2e sembra meno comprensibile di quella di TikZ agli occhi di chi non conosca le basi dei linguaggi dei due pacchetti esaminati ma TikZ richiede che sia specificato se un percorso sarà rettilineo o meno. Ovviamente il codice di Beccari ha anche il vantaggio di essere "parametrico", cioè di permettere la variazione delle dimensioni e del colore, cosa fattibile anche in TikZ grazie alle funzioni matematiche offerte da PGF. Nonostante le differenze progettuali, abbiamo proceduto a una comparazione più significativa e riportato nella tabella 1 il risultato di alcune misurazioni oggettive. Tali misurazioni riguardano la lunghezza del codice (ogni ambiente di disegno è stato incluso in un documento standalone identico), il suo numero di righe, la dimensione del PDF generato e la velocità di compilazione (media di dieci compilazioni con pdfLaTeX cronometrate con `time` di `bash`; i singoli valori rilevati sono riportati nella tabella 2).

Figura 1: Logo della casa editrice CLUT realizzato sia con TikZ, sia con curve2e.

Tabella 2: Tempi rilevati per le compilazioni dei logo. Da notare il valore più elevato delle prime compilazioni, quando per la prima volta si carica l'intero codice (anche di `time` nella memoria cache nel caso della prima compilazione).

TikZ (c.)	TikZ (s.)	curve2e
1,861	0,493	0,337
0,205	0,197	0,179
0,201	0,203	0,179
0,206	0,200	0,181
0,204	0,199	0,181
0,202	0,202	0,181
0,202	0,201	0,179
0,202	0,197	0,183
0,204	0,199	0,179
0,207	0,199	0,178

Alla fine del confronto, possiamo affermare che curve2e è preferibile rispetto a Ti*k*Z (il cui codice pure riesce a essere piú compatto) almeno nei casi come quello oggetto della sfida, in cui non si debba fare ricorso a strumenti esclusivi di Ti*k*Z e a maggior ragione se tali disegni sono molti all'interno del documento. Forse imparare il linguaggio di picture, il "capostipite" di pict2e di cui curve2e è un'estensione, non è banale, ma nella sua apparente semplicità la sintassi di Ti*k*Z non è meno facile.

Ringraziamenti

Gli autori della comunicazione ringraziano la CLUT nella persona dell'architetto Michele Ruffino per l'amichevole concessione a usare il logo di proprietà della casa editrice. Ringraziano inoltre gli anonimi *referee* per le puntuali osservazioni e suggerimenti che hanno migliorato la qualità dell'articolo e l'uniformità dei risultati mostrati.

Riferimenti bibliografici

Beccari, Claudio (2020). «The curve2e manual». CTAN. Leggibile col comando `texdoc curve2e-manual`.

Logo Foundation (2020). «Logo programming language». https://el.media.mit.edu/logo-foundation/what_is_logo/logo_programming.html. Ultima visita 16-5-2020.

Tantau, Till (2020). *Ti*k*Z & PGF Manual for Version 3.1.5b*. Leggibile col comando `texdoc tikz`.

▷ Claudio Beccari
claudio dot beccari at gmail dot com

▷ Gianluca Pignalberi
g dot pignalberi at gmail dot com

Questa rivista è stata prodotta
dal Gruppo Utilizzatori Italiani di TEX
usando esclusivamente software libero.

Versione elettronica per la diffusione via web.

Diciassettesimo convegno nazionale su TEX, LATEX e tipografia digitale

Trieste

Call for Papers

Trieste è stata scelta come sede per il diciassettesimo Convegno annuale su TEX, LATEX e tipografia digitale organizzato dal Gruppo Utilizzatori Italiani di TEX. Il Convegno sarà un momento di ritrovo e di confronto per la comunità LATEX italiana, tramite una serie di interventi atti sia a contribuire all'arricchimento sia a supportarne lo sviluppo.
Attenzione: per effetto delle disposizioni di sicurezza circa l'epidemia di Coronavirus, l'evento potrebbe essere tenuto per via telematica o, nella peggiore delle ipotesi, essere annullato.
Maggiori informazioni sul Convegno e sulle modalità di presentazione degli interventi saranno disponibili all'indirizzo:

`https://www.guitex.org/home/guit-meeting-2020/`

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 29, Aprile 2020

00029
9 771828 236001