I calcoli matematici in pdftex

Claudio Beccari

Sommario

Il motore di composizione `pdftex` così come i motori `xetex` e `luatex` possono fare calcoli ma né il primo, né gli altri compilatori sono stati pensati per farne di diversi da quelli relativamente semplici necessari per eseguire la composizione tipografica. Questo articolo cerca di esporre capacità e limiti delle funzionalità di calcolo ed indica alcuni metodi di calcolo numerico che differiscono non poco da quelli normalmente indicati nei testi di calcolo numerico.

Abstract

The typesetting engine `pdftex`, as well as the other typesetting engines `xetex` and `luatex` are capable of some computation functionalities different from those often simple ones necessary to the typesetting requirements. This tutorial is intended to show such computing capabilities and their limits; it also shows some numerical methods that are rather different from those normally described in textbooks on numerical computation.

1 Introduzione

Che un programma per calcolatore sia in grado di fare calcoli non dovrebbe stupire nessuno. Che il programma iniziale creato da Knuth attorno al 1978 facesse qualche calcolo pare del tutto evidente.

La cosa strana per un programma per elaboratore nel 1978, e ancora per diversi anni, era che ogni fabbricante di *mainframe* (perché allora i PC e gli Apple non esistevano ancora) non seguiva nessuno standard o per lo meno seguiva i propri.

Non sto a ricordare le date in cui sono stati introdotti alcuni standard; posso dire che il primo mainframe su cui ho messo le mani codificava i suoi dati in parole che oggi potremmo definire di 40 bit e usava una codifica dei numeri in decimale (o piuttosto in biquinario) codificato dove i quattro bit di ogni quaterna avevano i pesi 5, 4, 2, 1. Invero correva l'anno 1964 e nei quattordici anni prima della nascita di TEX78 erano successe molte cose. Tuttavia ho avuto occasione di lavorare su macchine in cui i byte potevano essere di 7 o di 8 o di 9 bit.

Knuth aveva una giusta preoccupazione: quella che il suo programma avrebbe dovuto girare su quasi tutte le macchine più diffuse; per questo era necessario limitarsi alle operazioni con i numeri interi; anche questi d'altra parte erano codificati in "big-endian" o "little-endian", in parole di 16 oppure 32 bit. Insomma anche nel 1978 le cose non erano stabilizzate.

In questo modo tutta l'aritmetica interna di ogni programma di compilazione del sistema TEX ricorre ai numeri interi; fu una decisione saggia? Sotto certi aspetti fu una decisione molto saggia; secondo la mia opinione personale oggi non lo è più, tanto che non solo `luatex` può fare calcoli in *floating point* (a virgola mobile, come diremmo in italiano) ricorrendo al linguaggio Lua, ma ogni distribuzione relativamente recente del sistema TEX contiene molti pacchetti in grado di fare calcoli con numeri fratti o floating point (pacchetto `fp`, Mehlich (1999)). Nello sviluppo del futuro LATEX3 esiste già un modulo `l3fp` per svolgere calcoli con numeri floating point e con rappresentazione scientifica. Molti pacchetti contengono le proprie librerie di macro per fare calcoli con numeri fratti, ma spesso sono costretti a ricorrere alle operazioni elementari predisposte da Knuth stesso fin dalle prime versioni del suo programma `tex`; questo continua ancora oggi ad essere la pietra di paragone con cui confrontare qualunque altro motore di composizione perché abbia il diritto di contenere la stringa `tex` nel suo nome.

Per conservare valori numerici Knuth aveva essenzialmente preso in considerazione due entità: i numeri e le lunghezze; queste ultime sono poi separate in lunghezze rigide e lunghezze elastiche, ma la differenza era solo che le seconde erano dotate anche di due "apposizioni" costituite dall'ammontare della contrattilità e della allungabilità; agli effetti di questo articolo ci interessano gli interi e le lunghezze rigide.

Questi due tipi di quantità numeriche sono conservabili in due tipi di registri diversi, quelli numerici e quelli dimensionali; i primi sono detti contatori; i secondi non hanno un nome solo, ma nel gergo di TEX sono chiamati *dimen register*. Entrambi questi tipi di registri sono "parole" di 4 byte di 8 bit, quindi di 32 bit; entrambi possono contenere quantità positive e negative; i registri dimensionali usano un altro bit per distinguere certe lunghezze speciali.

Ne segue che in valore assoluto un contatore può contenere un numero intero non superiore a $2^{31} - 1 = 2\,147\,483\,647$ che sembra sufficientemente grande da poter soddisfare ogni esigenza di calcolo "tipografico".

I registri dimensionali contengono le lunghezze espresse in *scaled point*; uno scaled point è la parte $1/2^{16}$ di un punto tipografico e viene conservato

nel registro sempre come numero intero di scaled point. Tenuto conto dei due bit riservati al segno e alle altre informazioni, il massimo numero di scaled point che possono essere inseriti in un registro dimensionale è $2^{30} - 1 = 1\,073\,741\,823$; questo valore corrisponde in realtà solo a circa 16 383.999 98 pt che a sua volta corrisponde a circa 5758.317 mm, vale a dire poco meno di sei metri. Anche questa è una grandezza sufficientemente grande per il calcolo tipografico, ma non è grandissima. Per evitare sorprese di overflow il valore massimo di 16 383.999 98 pt viene conservato nel registro dimensionale identificato con il nome molto esplicito `\maxdimen`; TEX considera questo valore una specie di "infinito" in un insieme di numeri finito.

I due tipi di registri possono scambiarsi i loro contenuti; nel senso che entrambi contengono solo numeri interi e quelle file di bit possono quasi sempre essere trasferite da un tipo di registro all'altro; il quasi è d'obbligo, perché i più grandi numeri interi hanno 31 bit contro 30 bit delle lunghezze.

Non solo, ma né l'un tipo di registro né l'altro accetta numeri fratti. In realtà i registri dimensionali possono essere visti sia come valori interi di scaled point, sia come valori fratti di punti tipografici con la "virgola", la quale separa la parte intera dalla parte fratta ed ha alla sua destra 16 posizioni binarie fisse. I registri lunghezza possono essere quindi visti come numeri a virgola fissa. Ma sedici cifre binarie corrispondono a circa 4.8 cifre decimali, quindi trasformando i contenuti dei registri in valori leggibili dalle persone otteniamo numeri decimali con poco meno di 5 cifre decimali corrette.

Fin dall'inizio del sistema TEX le operazioni eseguibili sui contenuti dei registri numerici o dimensionali potevano essere solo la somma algebrica, la moltiplicazione e la *divisione intera*. Questo va sottolineato: la divisione intera restituisce solo la parte intera del quoziente, non fornisce un quoziente fratto.

Certo una macchina calcolatrice con funzionalità limitate dalla natura interna dei suoi operandi non è granché; anzi direi che si tratta di una calcolatrice molto modesta.

Va detto inoltre che i soli numeri fratti accettati dai motori di composizione sono i fattori moltiplicativi delle lunghezze. Questi sono usati spessissimo senza fare particolare attenzione al fatto che si stia usando una specie di eccezione alla regola; per esempio, sono usati quando durante l'importazione di una figura viene specificato `width=0.7\textwidth` fra le opzioni di `\includegraphics`.

2 Le operazioni di base

Le operazioni di base di tutti i motori di composizione accettano solo `\advance` per eseguire una somma algebrica, `\multiply` per eseguire una moltiplicazione, `\divide` per eseguire una divisione intera.

Gli operandi per `\advance` dovrebbero essere della stessa natura, o entrambi numeri o entrambi lunghezze; il risultato dell'operazione viene messo nello stesso registro del primo operando che viene perciò sovrascritto e il contenuto precedente viene quindi "perso". Tipico è il modo con cui viene definita la larghezza del margine sinistro:

```
\@tempdima= \paperwidth
\advance\@tempdima by -\textwidth
\oddsidemargin =0.4\@tempdima
\advance\oddsidemargin by -1in
```

Come si vede in questo semplice esempio, singoli comandi come `\advance` non accettano espressioni, ma svolgono una operazione alla volta; su questo punto si tornerà più avanti.

Il codice mostrato sopra usa la sintassi nativa di TEX, ma con LATEX sarebbe possibile usare i comandi specifici di quel markup come `\setlength` e `\addtolength`; cambierebbe la forma ma non la sostanza.

Un altro esempio è quello con cui si fissa l'altezza della gabbia del testo affinché contenga un numero intero di righe di testo composto con il carattere di default nel corpo normale:

```
\@tempdima=\paperheight
\advance \@tempdima by -2in
\advance\@tempdima by -1.5in
\divide\@tempdima by \baselineskip
\@tempcnta=\@tempdima
\textheight=\@tempcnta\baselineskip
\advance\textheight by \topskip
```

In questo caso vengono sottratti all'altezza della pagina i margini superiore e inferiore e lo spazio (forfettario) da destinare alla testatina e al piedino incluse le distanze di separazione dal testo vero e proprio. Lo spazio rimanente viene diviso per lo scartamento del font normale e il risultato viene poi assegnato ad un contatore. Infine l'altezza del testo viene ottenuta moltiplicando lo scartamento (`\baselineskip`) per il numero intero ottenuto dalla divisione intera eseguita con `\divide`.[1]

Questi sono gli usi tipografici delle operazioni fra numeri e lunghezze; pur nella loro modestia sono più che sufficienti per svolgere le funzioni tipografiche richieste dai motori di composizione. Certamente non sono sufficienti per fare calcoli su numeri fratti.

3 L'estensione e-TEX

Nel 2005 le estensioni introdotte dal *working group* per lo sviluppo del New Typesetting System sono

1. L'esempio è ispirato al codice tratto dal file di opzione per il corpo normale di 10 pt della classe `book`; sono stati sostituiti solo i comandi LATEX `\setlength` e `\addtolength` con i comandi nativi di TEX.

state introdotte in tutti i motori di composizione, tranne nell'eseguibile knuthiano "puro" `tex`.

Uno dei pochi pacchetti che immediatamente fece uso di queste nuove funzionalità fu proprio la classe arstexnica, quella usata per comporre questa rivista; ma non tutte le distribuzioni del sistema TEX furono altrettanto pronte ad incorporare queste nuove funzionalità e si verificarono piccoli inconvenienti di incompatibilità; ma dal 2006 la distribuzione di riferimento di ogni sistema TEX è la TeXLive e da allora non ci sono più stati inconvenienti.

Le nuove funzionalità sono molteplici e si manifestano in diversi ambiti; qui ci interessano le possibilità di calcolo, in particolare i comandi `\numexpr` e, specialmente, `\dimexpr` che permettono di svolgere calcoli ricorrendo ad espressioni matematiche numeriche, dimensionali e miste, cosa che con i comandi primitivi del sistema knuthiano non era possibile.

Tanto per fare un esempio, la determinazione dell'altezza delle gabbia interna `\textheight`, con le nuove funzionalità diventa:

```
\textheight=\dimexpr(\paperheight-2in
   -1.5in)/\baselineskip*\baselineskip
   +\topskip\relax
```

Le nuove funzionalità, oltre alla possibilità di eseguire le quattro operazioni, comportano due novità:

- le divisioni sono sempre divisioni intere, ma il risultato viene arrotondato, non troncato come avviene con i precedenti comandi nativi;
- l'operazione di scalamento avviene conservando il risultato intermedio in un registro di 64 bit, il doppio di un un normale registro da 32 bit.

L'operazione di scalamento va meglio spiegata con un esempio. Supponiamo di voler avere una gabbia di stampa con le stesse proporzioni della pagina e supponiamo di conoscere o di avere già specificato la giustezza del testo `\textwidth`. Dobbiamo determinare `\textheight` in modo che il rapporto altezza su larghezza della gabbia sia uguale al rapporto altezza su larghezza della pagina; dobbiamo cioè determinare

```
\textheight= \paperheight/\paperwidth*\textwidth
```

Semplice e intuitivo; scriviamo infatti la riga:

```
\textheight=\dimexpr
 \paperheight*\texwidth/\paperwidth\relax
```

Che cosa succede specificando prima la moltiplicazione e poi la divisione? I due fattori della moltiplicazione sono lunghezze codificate con "parole" di 32 bit le quali contengono i valori delle lunghezze espressi con un numero intero di scaled point; il risultato della moltiplicazione richiede quindi 64 bit per contenere qualunque possibile valore, grande o piccolo, senza doverlo troncare o arrotondare. La successiva divisione per una lunghezza, anch'essa contenuta in una parola di 32 bit, produce un quoziente intero di 32 bit che eventualmente viene arrotondato, invece che troncato; perciò il calcolo dà luogo ad una stringa di bit che rappresentano correttamente una lunghezza in scaled point, ovvero una lunghezza in punti tipografici con 16 cifre binarie fratte.

Cosa distingue questo modo di procedere da quello che si potrebbe ottenere con i comandi primitivi descritti nel paragrafo precedente? Nel secondo caso la prima moltiplicazione produrrebbe un overflow. Infatti una altezza della carta di 297 mm, corrisponde a 845 pt; una larghezza della gabbia, diciamo, di 160 mm corrisponde a 455 pt e il prodotto darebbe luogo ad un valore di 384 475 assai maggiore di `\maxdimen`; quindi l'unica cosa che si otterrebbe è un messaggio di overflow e l'arresto della compilazione.

Cosa succederebbe allora se si eseguisse prima la divisione? Se la larghezza della carta è di 210 mm, che equivale a 597 pt, la divisione intera dell'altezza per la larghezza darebbe il valore $(\mathrm{INT}(845/597) \approx \mathrm{INT}(1.414) = 1$, e si perderebbe ogni informazione utile.

Ecco, questo è un punto chiave; gli scalamenti vanno sempre fatti usando questa nuova funzionalità di e-TEX.

4 Alcuni pacchetti che consentono di fare calcoli matematici

Visto che i motori di composizione sono così modesti nel fare i calcoli, già da molto tempo esistono pacchetti che consentono di farli fare al programma di composizione. Tutti questi pacchetti in qualche modo, diretto o indiretto, ricorrono al trucco seguente: ogni numero fratto viene usato come moltiplicatore della lunghezza predefinita `\p@` che di suo è lunga un punto tipografico; su questa ed altre lunghezze ottenute nello stesso modo si eseguono le operazioni matematiche che si desiderano; il risultato finale viene "mostrato" in forma decimale, dopo avergli tolto le unità di misura. Per esempio, volendo sommare 1,333333 e 2,66666, si potrebbe procedere così:

```
\@tempdima=1.333333\p@
\@tempdimb=2.666666\p@
\edef\@tempA{%
\strip@pt\dimexpr\@tempdima+\@tempdimb}%
\texttt{\@tempA}
```

Il codice di questo esempio produce il risultato seguente: `4`. In questo caso il risultato è tondo e il fatto che esso non sia `3.999999` deriva dall'arrotondamento sempre eseguito dall'aritmetica delle espressioni matematiche di e-TEX.

Il pacchetto principale noto da molti anni per svolgere questi calcoli è il pacchetto calc (Thorup *et al.*, 2014). calc non svolge solo le quattro operazioni, ma ridefinisce anche tutti i comandi LaTeX che accettano numeri o lunghezze, in modo che siano in grado di analizzare le espressioni matematiche. Siccome molti altri comandi LaTeX fanno uso dei comandi ridefiniti, molti altri comandi possono beneficiare delle espressioni che calc è in grado di gestire. Si noti inoltre che per usare numeri reali come fattori nelle moltiplicazioni e nelle divisioni bisogna racchiuderli negli argomenti di `\real` oppure `\ratio` in modo da poterli usare convenientemente nelle espressioni matematiche.

Diversi altri pacchetti usano loro macro interne per eseguire i calcoli; per esempio l'estensione pict2e (Gässlein *et al.*, 2014), già documentata da Leslie Lamport nella sua seconda edizione del manuale di LaTeX del 1994 ma non effettivamente resa disponibile fino almeno al 2003, ha la sua collezione di macro necessaria per realizzare le varie estensioni annunciate circa 10 anni prima.

I pacchetti pgf, tikz (Tantau, 2013) e pgfplots (Feuersänger, 2015) dispongono di loro macro interne e di librerie complete per il calcolo di numerose funzioni matematiche; anzi, pgfplots contiene delle raffinate librerie matematiche che consentono di calcolare le funzioni matematiche in modo decimale, con un numero elevato di cifre significative; esse fanno uso del pacchetto fp, per eseguire calcoli in floating point. Certamente per usare questi metodi di calcolo bisogna rinunciare alla velocità di esecuzione e si corre anche il rischio che la memoria principale del programma di compilazione risulti insufficiente. Il problema è noto e nei vari forum, compreso quello del GuIt, ci sono diversi interventi a questo proposito.

Molto prima di questi pacchetti, nati essenzialmente dopo l'avvento di LaTeX 2_ε, il pacchetto pictex (Wichura, 1988) permetteva a Plain TeX di cimentarsi con il disegno di diagrammi, anche semi- o bi-logaritmici, essendo esso in grado di calcolare i logaritmi decimali con sufficiente precisione.

Esistono altri pacchetti in grado di fare calcoli. Sebbene sia poco noto e poco usato, mi piace ricordare il pacchetto curve2e (Beccari, 2015) che estende ulteriormente il pacchetto pict2e impiegando in lungo e in largo calcoli con i numeri non solo fratti ma anche complessi e quindi usando coppie ordinate di numeri fratti. Per gestire i numeri complessi pict2e usa anche la distanza pitagorica (vedi più avanti) e l'anomalia su quattro quadranti; inoltre le coordinate polari (per altro comunemente usate anche da tikz e pgfplots. Nell'articolo (Beccari, 2013) ci sono anche altri spunti per eseguire calcoli con `pdftex`.

Posso citare i pacchetti picture (Oberdiek, 2009) e xpicture (Fuster, 2012). Uno estende l'ambiente picture di LaTeX, già esteso da pict2e, accettando coordinate espresse sia nell'unità di misura `\unitlength` tipico dell'ambiente picture, sia le unità assolute riconosciute da ogni motore di composizione. L'altro estende le funzionalità disegnando in modo semplice curve normalmente disegnate nei corsi di geometria analitica nelle scuole secondarie superiori, compresi i diagrammi di polinomi e le curve parametriche; esso ricorre alle spline quadratiche di Bézier, che a loro volta richiedono calcoli con numeri fratti.

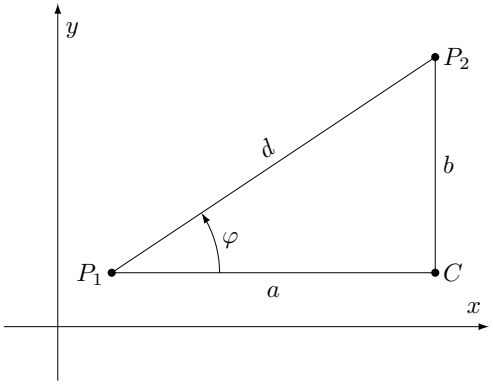

Figura 1: La distanza fra due punti

5 La distanza pitagorica e la radice quadrata

Se non è già definita da macro interne, la distanza fra due punti qualsiasi nel piano (x, y) richiede l'uso del teorema di Pitagora per calcolare la lunghezza dell'ipotenusa di un triangolo rettangolo, figura 1.

Date le coordinate di P_1 e P_2, non è un problema determinare le lunghezze a e b dei due cateti del triangolo rettangolo della figura 1; si tratta solo delle differenze con segno fra le ascisse o, rispettivamente, le ordinate dei due punti. Sono due numeri fratti con segno, segno che però ai fini del calcolo della distanza d dei due punti è irrilevante. Sia $|a| > |b|$, come rappresentato nella figura; ma senza formalizzare sui simboli, sia $|a|$ la lunghezza del maggiore dei due cateti. È noto allora che la distanza d è data da

$$d = \sqrt{a^2 + b^2} \tag{1}$$

Peccato che non esiste nessun comando nativo di TeX che permetta di calcolare quella radice quadrata direttamente e che sia necessario ricorrere ad una macro che permetta di determinarla. Il noto metodo di Newton procede iterativamente; si sceglie un numero di partenza "arbitrario" e si divide il radicando per questo numero; quindi si determina un nuovo numero prendendo la media fra il rapporto appena trovato e il numero usato come divisore. È evidente che una volta trovato il numero giusto rappresentante la radice quadrata, il rapporto e il numero sono uguali e il problema è risolto.

Numericamente parlando dobbiamo ricordare che abbiamo a che fare con una macchina calcolatrice piuttosto scadente e che dobbiamo scegliere un numero conveniente per iniziare il calcolo iterativo; dobbiamo inoltre trovare un test adeguato per decidere quando il risultato è stato trovato.

Per fare questo è necessario riscrivere l'equazione 1 in modo conveniente restringendo il dominio del radicando. Supponiamo, come detto sopra, che sia $|a| > |b|$:

$$d = |a|\sqrt{1 + x^2} \tag{2a}$$

dove

$$x = \frac{b}{a} \tag{2b}$$

In questo modo siamo sicuri che $x < 1$ e che il radicando sia minore di 2; tuttavia se a o b fosse nullo dovremmo procedere diversamente. Il codice diventa pertanto il seguente:

```
\def\distanza#1,#2to#3{%
\def\t@X{#1}\def\t@Y{#2}%
\@tempdima=\t@X\p@
\ifdim\@tempdima<\z@
    \@tempdima=-\@tempdima\fi
\@tempdimb=\t@Y\p@
\ifdim\@tempdimb<\z@
    \@tempdimb=-\@tempdimb\fi
\ifdim\@tempdima=\z@
    \ifdim\@tempdimb=\z@
        \def\@X{0}\@tempdimc=\z@
    \else
        \def\@X{0}\@tempdimc=\@tempdimb
    \fi
\else
  \ifdim\@tempdima>\@tempdimb
    \DividE\@tempdimb by\@tempdima to\@X
    \@tempdimc=\@tempdima
  \else
    \DividE\@tempdima by\@tempdimb to\@X
    \@tempdimc=\@tempdimb
  \fi
\fi
%-------------
\unless\ifdim\@tempdimc=\z@
  \unless\ifdim\@X\p@=\z@
    \@tempdima=\@X\p@
    \@tempdima=\@X\@tempdima
    \advance\@tempdima\p@%
    \@tempdimb=\p@%
    \@tempcnta=5\relax
    \@whilenum\@tempcnta>\z@\do{%
    \DividE\@tempdima by\@tempdimb to\@X
    \advance\@tempdimb \@X\p@
    \@tempdimb=.5\@tempdimb
    \advance\@tempcnta\m@ne}%
    \@tempdimc=\@X\@tempdimc
  \fi
\fi
\edef#3{\strip@pt\@tempdimc}%
\ignorespaces}%
```

La macro `\distanza` è ad argomenti delimitati; i primi due, separati da una virgola, sono due numeri fratti cioè le misure dei due cateti nelle unità dell'ambiente picture; il terzo argomento è una control sequence a cui verrà assegnato il risultato e, in quanto rappresentato da un unico token, non necessita di essere racchiuso fra graffe.

Prima della linea di commento vengono eliminati i casi ovvi nei quali uno o l'altro cateto è di lunghezza nulla; viene poi deciso quale sia il cateto maggiore e viene costruito il rapporto chiamato con la macro `\@X`.

Nella seconda parte della macro viene costruito nel registro dimensionale `\@tempdima` il radicando; nel registro `\tempdimb` viene posta la lunghezza unitaria da usare come valore iniziale delle iterazioni; successivamente viene imposto il contatore `\@tempcnta` con il numero totale delle iterazioni da eseguire sotto controllo del comando del nucleo di LaTeX `\@whilenum`. L'unica macro non ancora descritta è `\DividE` ad argomenti delimitati; non è così misteriosa, ma semplicemente si appoggia ad una espressione dimensionale e definisce il terzo argomento (una control sequence) con il valore numerico del quoziente. Quindi

```
\DividE\@tempdima by\@tempdimb to\@X
```

significa "dividi il radicando per l'approssimazione corrente della radice e assegnalo alla macro `\@X`" (il nome è lo stesso, ma la precedente definizione di rapporto dei cateti a questo punto non serve più).

Si noti che questo semplice calcolo diventa molto utile se si vogliono ruotare oggetti senza ricorrere a pacchetti esterni. Infatti i rapporti a/d e b/d fra i cateti e l'ipotenusa del triangolo rettangolo della figura 1 non sono altro che il coseno e il seno dell'angolo φ. Non è quindi difficile scrivere direttamente la matrice di trasformazione in linguaggio PDF che richiede appunto i seni e i coseni con i segni giusti per eseguire la trasformazione affine costituita dalla rotazione; di nuovo si tratta di eseguire due divisioni fra numeri fratti trasformati in lunghezze e sistemare i due rapporti nelle posizioni giuste della matrice di trasformazione. Visto che questo discorso può apparite fumoso senza un esempio, ecco almeno la sintassi[2]:

2. Qui si usa la sintassi del linguaggio PDF; per usare il linguaggio PostScript le cose sono molto simili, ma le parole chiave, ridotte a una o due lettere con il linguaggio PDF, sono parole complete nel linguaggio PostScript. Ecco perché nel titolo dell'articolo è stato specificato che si sarebbe discussa la possibilità di eseguire calcoli con il motore di composizione `pdftex`. Invece con `luatex` non è necessario fare nulla di speciale perché il linguaggio Lua può svolgere tutti i calcoli necessari; con `xetex` chi produce il file pdf è una versione modificata del programma `dvipdfm` che richiederebbe un'altra variante ancora.

```
% Salvare la matrice corrente
\special{pdf: q}
% Impostare la matrice affine
\special{pdf: ⟨cos φ⟩ ⟨sin φ⟩ ⟨− sin φ⟩ ⟨cos φ⟩
⟨0⟩ ⟨0⟩ cm}
⟨Oggetto da ruotare⟩
% Ripristinare la matrice precedente
\special{pdf: Q}
```

Ovviamente si può definire una macro che faccia tutto il necessario senza dover consultare questa sintassi ogni volta. Ma qui questa possibile applicazione è segnalata solo per mostrare come spesso si possono usare calcoli interni svolti dallo stesso motore di composizione.

6 Come usare i registri senza allocarli

Nelle macro esposte sono stati usati registri temporanei già allocati, tutti quelli i cui nomi contengono la stringa `temp`. Nel fare altri conti serve poter usare tanti registri senza doversi preoccupare se siano già stati definiti. Un tipico esempio è quello della definizione di `\DividE`:

```
\def\DividE#1by#2to#3{\bgroup
  \dimendef\Num2254\relax
  \dimendef\Den2252\relax
  \dimendef\@DimA 2250
  \Num=\p@ \Den=#2\relax
  \ifdim\Den=\z@
    \edef\x{\noexpand\endgroup\noexpand
    \def\noexpand#3{\strip@pt\maxdimen}}%
  \else
    \@DimA=#1\relax
    \edef\x{%
      \noexpand\egroup\noexpand
        \def\noexpand#3{\strip@pt
        \dimexpr\@DimA*\Num/\Den\relax}}%
  \fi
\x\ignorespaces}%
```

In questa macro ad argomenti delimitati, il primo argomento è il dividendo costituito da una lunghezza; il secondo è un altra lunghezza che costituisce il divisore; il terzo è una macro a cui viene assegnato il numero fratto costituito dal quoziente; le prime due lunghezze è bene che siano conservate in registri dimensionali, ma a questo punto (e in molti altri punti...) è meglio non fare riferimento ai registri predefiniti, ma usare registri "qualsiasi" associando loro un nome di comodo. Qui il nome `\Num` è associato al registro dimensionale 2254 e il nome `\Den` al registro dimensionale 2252; i due numeri sono scelti "a caso" fra le migliaia di registri accessibili con le estensioni di e-TEX, che permettono di arrivare ad usarne più di 32 000. Questi, d'altra parte, potrebbero essere usati anche da altri pacchetti, quindi è meglio eseguire queste associazioni dentro un gruppo, farvi dentro tutti i calcoli e poi usare uno dei "dirty tricks" (sporchi trucchi), descritti da Knuth nell'apposita appendice del TEXbook, (Knuth, 1984, app. D), per portare il risultato fuori dal gruppo.

La macro comincia con un `\bgroup` che apre un gruppo con una graffa implicita. Con il comando nativo di TEX `\dimendef` associamo un nome ad un registro dimensionale; carichiamo poi il registro che identifica il numeratore con la lunghezza unitaria e il registro che identifica il denominatore con la lunghezza che costituisce il divisore; siccome bisogna essere sicuri di non dividere per zero, eseguiamo un test sul denominatore; se questo è nullo possiamo decidere, e qui lo si fa, di non impostare un segnale di errore fatale, ma di associare una specie di valore infinito al quoziente; questo viene fatto mediante la definizione di `\x` con `\maxdimen` (sulla quale torniamo fra poco).

Altrimenti si assegna il dividendo al registro dimensionale `\@DimA` e si calcola il quoziente definendo poi la macro `\x` con il suo valore fratto privato dell'unità di misura. Si noti che l'esecuzione della definizione "espansa" mediante `\edef`, esegue a sua volta tutte le macro contenute nel suo testo sostitutivo, tranne quelle precedute da `\noexpand`; quindi del suo testo sostitutivo viene calcolata solo l'espressione dimensionale sotto forma di scalamento e poi vengono tolti i pt dal risultato. Eseguita questa definizione espansa, si esegue la macro `\x` che per prima cosa chiude il gruppo con `\egroup`, cancellando la sua stessa definizione dalla memoria e ripristinando tutti i valori precedenti contenuti nei registri usati dentro il gruppo; tuttavia anche se la macro `\x` non esiste più quello che si sta eseguendo è il suo testo sostitutivo e quindi il valore del quoziente viene assegnato alla macro che costituisce il terzo argomento di `\DividE`.

Questo modo di operare è particolarmente indicato per eseguire i calcoli. Anche se non sempre è necessario ricorrere alla macro `\x` che cancella se stessa, altrettanto spesso lo è; si evitano così conflitti con altre macro usate nello stesso documento e con le macro usate in altri pacchetti.

7 Trasformare i punti in millimetri

Ecco un piccolo esempio che a me pare abbastanza utile. Si tratta di una piccola macro che trasforma le lunghezze espresse in punti in lunghezze espresse in millimetri. Nonostante il rapporto fra millimetro e punto tipografico sia quasi uguale a 3 (per la precisione a 2,84528), passare dai millimetri ai punti moltiplicando per 3 e viceversa dai punti ai millimetri dividendo per 3, almeno per avere una idea delle grandezze e sia pure con un errore di circa il 5%, non è una cosa impossibile a mente: tuttavia qualche volta occorre un valore più preciso. Perché non farlo calcolare al motore di composizione?

Allora ricordiamo che il punto tipografico usato dal sistema TEX è quello anglosassone che si rapporta al pollice mediante la definizione per la quale

un pollice vale esattamente 72,27 pt; d'altra parte un pollice vale 25,4 mm, quindi queste due costanti vanno usate in modo da semplificare i pollici. Ecco dunque la macro per scrivere una lunghezza in millimetri:

```
\newcommand\pttomm[1]{\bgroup
\dimendef\mmtoinch 3333
\dimendef\pttoinch 3334
\dimendef\inlength 3335
\mmtoinch=25.4\p@ \pttoinch=72.27\p@
\inlength=#1\relax
\edef\x{\noexpand\egroup\strip@pt
\dimexpr\inlength*\mmtoinch/\pttoinch
{\noexpand\,}mm}\x}
```

Con questi comandi la larghezza di pagina di questo testo vale 209.99994 mm. Come si vede gli arrotondamenti non proteggono da una diminuzione di precisione dei comandi usati, ma sicuramente nessuno si lamenterebbe di leggere 209,99994 mm, che si capisce al volo, piuttosto che 598 pt che ci lascia indifferenti, visto che non abbiamo una consuetudine giornaliera con i punti tipografici.

8 Le funzioni trigonometriche

Le funzioni trigonometriche sono già codificate nel pacchetto trig (Carlisle, 1999), caricato di default dal pacchetto graphicx (Carlisle, 2014). Non sono male ma ritengo che sia meglio rifarsi alle formule parametriche della trigonometria:

$$\sin\theta = \frac{2}{\cot x + \tan x} \tag{3a}$$

$$\cos\theta = \frac{\cot x - \tan x}{\cot x + \tan x} \tag{3b}$$

$$\tan\theta = \frac{2}{\cot x - \tan x} \tag{3c}$$

dove θ è l'angolo in gradi e $x = \theta/114.591559$ è metà dello stesso angolo in radianti.

Perché i gradi? Semplice: i gradi contenuti in un angolo giro o piatto o retto sono un numero intero ed è quindi molto più semplice ridurre l'angolo di cui calcolare la funzione trigonometrica ad un intervallo principale. Inoltre la tangente ha uno sviluppo in frazione continua:

$$\tan x = \cfrac{1}{\frac{1}{x} - \cfrac{1}{\frac{3}{x} - \cfrac{1}{\frac{5}{x} - \cfrac{1}{\frac{7}{x} - \cfrac{1}{\frac{9}{x} - \cfrac{1}{\frac{11}{x} - \dots}}}}}} \tag{4}$$

molto semplice da calcolare con una macchina calcolatrice modesta come quella di cui disponiamo e anche molto precisa se si limita il dominio principale all'intervallo $[-45°, +45°]$. Ecco quindi che le formule parametriche sono particolarmente indicate, anche se la forma espressa nelle formule 3 non è quella familiare che si trova in ogni libro delle scuole secondarie superiori.

Se il modulo dell'angolo in radianti non supera $\pi/4$, l'espressione 4 con una "calcolatrice" migliore potrebbe dare dieci o undici cifre esatte; noi dobbiamo accontentarci al massimo di cinque cifre dopo la virgola e siamo fortunati se ne otteniamo quattro.

Ora non si espone come usare la tangente del mezzo angolo per calcolare il seno, il coseno la tangente dell'angolo intero; si tratta di macro che si scrivono quasi immediatamente partendo dalle equazioni 3. Ma è opportuno che queste macro si occupino: di ridurre l'angolo intero all'intervallo fondamentale $[-\pi/2, +\pi/2]$; dei valori speciali, come i multipli interi di $\pi/2$; dei segni eventualmente necessari fino al momento di chiamare la macro della tangente del mezzo angolo la quale deve restituire direttamente sia il valore della tangente, sia quello della cotangente. Ecco quindi il codice di questa macro per il calcolo della tangente del mezzo angolo.

```
\def\g@tTanCotanFrom#1to#2and#3{%
\DividE 114.591559\p@ by#1to\X@
\@tdB=\X@\p@
\countdef\I=2546 \I=11 \def\Tan{0}%
\@whilenum\I>\z@\do{%
  \@tdC=\Tan\p@ \@tdD=\I\@tdB
  \advance\@tdD-\@tdC
  \DividE\p@ by\@tdD to\Tan
  \advance\I-2\relax}%
\def#2{\Tan}\DividE\p@ by\Tan\p@ to\Cot
\def#3{\Cot}\ignorespaces}%
```

Come si vede si tratta di una macro piuttosto breve e questo è dovuto alla possibilità di usare la frazione continua indicata dall'equazione 4. L'iterazione è controllata dal comando `\@whilenum`, facente parte del nucleo di LaTeX, ed inizia dall'assegnazione del valore 11 al contatore `\I`. Forse è una esagerazione, ma non guasta; di sicuro il tempo perso ad eseguire una o due iterazioni di troppo è trascurabile visto che tutti i calcoli sono sviluppati nell'unità centrale di elaborazione del calcolatore. Per la lettura del codice si tenga presente che `\X@` rappresenta l'inverso del mezzo angolo trasformato in radianti; le variabili `\@td...` devono essere preventivamente allocate per registri lunghezza, oppure associati precedentemente con registri lunghezza di numero "qualsiasi", purché le macro per il calcolo del seno, coseno e tangente provvedano correttamente a confinare queste associazioni dentro un gruppo. Questi sono trucchi già descritti in precedenza.

Non è da sottostimare la semplicità dell'uso della frazione continua; quando è possibile disporre di una frazione continua per approssimare una funzione trascendente, è conveniente usare la frazione

continua piuttosto che polinomi opportuni (generalmente polinomi di Chebyshchev) spesso usati per la determinazione delle funzioni trascendenti nel firmware di molti programmi di calcolo.

9 Le spline di Bézier

Le spline di Bézier sono curve parametriche che permettono di spostare un punto $P = (x, y)$ da un punto iniziale $P_0 = (x_0, y_0)$ ad un punto finale $P_1 = (x_1, y_1)$ lungo una traiettoria governata da espressioni polinomiali. Le più comuni sono le spline di primo, di secondo e di terzo grado; dipendono tutte dal parametro t di cui normalmente si considera solo l'intervallo $0 \le t \le 1$:

$$\mathcal{B}_1 = P_0(1-t) + P_1 t \tag{5a}$$

$$\mathcal{B}_2 = P_0(1-t)^2 + C2t(1-t) + P_2 t^2 \tag{5b}$$

$$\mathcal{B}_3 = P_0(1-t)^3 + C_a 3t(1-t)^2 + C_b 3t^2(1-t) + P_1 t^3 \tag{5c}$$

La spline di primo grado $\mathcal{B}_1$ è semplicemente un segmento che unisce i punti P_0 e P_1; chiaramente in questa come nelle altre spline, per $t = 0$ la funzione restituisce le coordinate di P_0 e per $t = 1$ essa restituisce le coordinate di P_1. Generalmente questa spline non è programmata come linea parametrica perché esistono metodi più diretti per tracciare segmenti di retta.

La spline di secondo grado $\mathcal{B}_2$ è un arco di parabola che unisce P_0 e P_1 con le tangenti nei due punti estremi che sono parallele ai segmenti $\overline{P_0C}$ e $\overline{CP_1}$; il punto C controlla quindi le direzioni delle tangenti di partenza e di arrivo; l'intera curva giace all'interno del *convex hull* delimitato dal triangolo P_0CP_1; per $t = 0{,}5$ la funzione fornisce un punto nella mezzeria della bisettrice dell'angolo $\widehat{P_0CP_1}$.

Le spline di Bézier di secondo ordine sono facili da usare perché se si specificano le direzioni, il punto di controllo è univocamente determinato; esse sono normalmente usate per descrivere i contorni dei caratteri TrueType e per alcuni caratteri OpenType.

Siccome gli archi delle spline di Bézier di secondo grado sono archi di parabole, non possono creare curve con flessi, a meno che il punto di unione di due archi consecutivi non venga fatto coincidere con il punto di flesso.

Le spline di Bezier di terzo grado sono le più versatili, nel senso che al variare di t fra zero e uno, il punto restituito dalla spline si muove lungo una cubica che può presentare anche un flesso; in ogni caso tutta la curva si svolge dentro il *convex hull* formato dal poligono con i vertici nei punti P_0, C_a, C_b, P_1. Le tangenti nei punti estremi sono parallele ai segmenti $\overline{P_0C_a}$ e $\overline{C_bP_1}$. Diversamente dalle curve di secondo grado, le spline di terzo grado non sono univocamente determinate dalle tangenti, perché i punti di controllo sono due e non uno solo. C'è quindi un margine di arbitrarietà che consente anche di controllare la curvatura dell'arco come è osservabile nella figura 2 dove, scegliendo opportunamente la posizione e la distanza dei punti di controllo dai punti estremi, si riesce a generare un arco indistinguibile ad occhio nudo da un arco di cerchio.

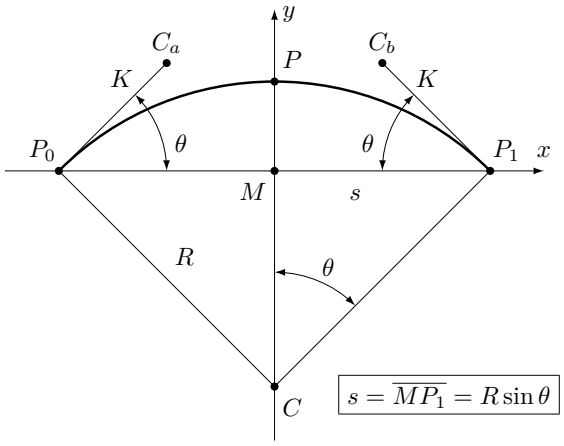

Figura 2: Estremi e punti di controllo di una curva di Bézier di terzo grado che approssima un arco di cerchio

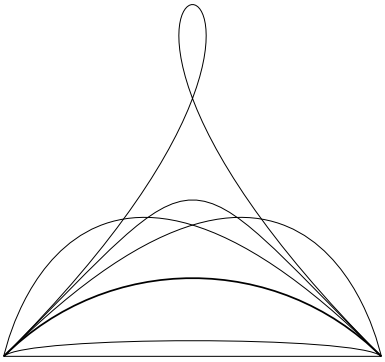

Figura 3: Diversi archi di spline di Bézier con gli stessi estremi ma con i punti di controllo via via più distanti dai punti estremi fino a formare un anello

Nella figura 3, invece, si sono scelti per i diversi archi gli stessi punti estremi e le stesse direzioni delle tangenti nei punti estremi, direzioni uguali a quelle dell'arco di cerchio disegnato più scuro; le distanze dei punti di controllo dai punti estremi sono state scelte uguali o diverse; nulle, medie o grandi; gli archi più bassi sono più tesi, fino a confondersi con la loro corda; gli archi superiori asimmetrici hanno o C_a o C_b coincidenti con l'estremo a cui sono associati, ma con l'altro punto di controllo non coincidente col relativo estremo; gli archi superiori simmetrici hanno le distanze dei punti di controllo crescenti fin quando la curva forma un anello.

Questo quindi sottolinea la grande versatilità delle curve di terzo grado, ma mette anche in risalto la difficoltà di scegliere opportunamente i punti di controllo.

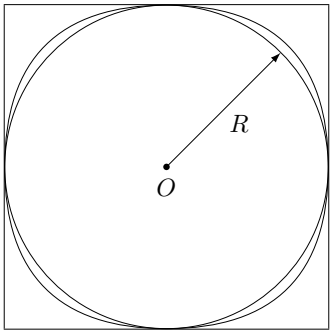

FIGURA 4: due "cerchi" disegnati uno con spline cubiche e uno con spline quadratiche

Fatta questa descrizione delle spline di Bézier, illustro una serie di calcoli che servono per fare una delle altre magie delle spline di terzo grado: disegnare una curva di secondo grado con una spline di terzo grado; questi calcoli devono essere svolti dal motore di composizione; so per certo che il pacchetto pict2e li svolge e che la sua estensione curve2e sfrutta le stesse macro. Non ho verificato se queste funzionalità e questi calcoli vengano svolti da altri pacchetti, in particolare da tikz, pgf, pgfplots o xpicture o da altri pacchetti che non ho mai usato, ma ai fini di questo articolo è irrilevante.

Il problema consiste nel determinare i punti di controllo C_a e C_b di una spline di terzo grado che collega i punti P_0 e P_1, in modo che la curva diventi esattamente una funzione di secondo grado del parametro t.

Bisogna prendere l'equazione 5c e imporre che i coefficienti di terzo grado si annullino e che quel che resta possa essere messo nella forma dell'equazione 5b. Si tratta di un semplice esercizio d'algebra elementare, ma non dovrebbe sorprendere che una curva di terzo grado possa avere i coefficienti scelti in modo tale da ridursi ad una curva di secondo grado; la soluzione è la seguente:

$$C_a = C + (P_0 - C)/3 \text{ e } C_b = C + (P_1 - C)/3 \quad (6)$$

In altre parole le ascisse e le ordinate dei punti C_a e C_b sono delle semplicissime combinazioni lineari delle corrispondenti coordinate dei punti estremi e di quelle dell'unico punto di controllo della spline di secondo grado; ecco: queste semplici combinazioni lineari si possono fare agevolmente con l'aritmetica descritta in questo articolo e sono alla portata di tutti, purché conoscano come trasformare un numero fratto in una lunghezza e, dopo averci lavorato sopra, sappiano come togliere la stringa `pt` dalla loro rappresentazione decimale.

Curiosità: benché le spline di Bézier di secondo grado siano usate in modo esclusivo per descrivere i contorni dei font TrueType, queste spline non sono così versatili come le spline di terzo grado; quindi per descrivere un dato contorno è necessario usare più archi quadratici di quanti ne siano necessari con gli archi cubici. Nella figura 4 sono mostrati due "cerchi", entrambi descritti mediante quattro archi di 90° ciascuno; un cerchio è descritto con le spline di terzo grado e, benché sia solo un'approssimazione di un vero cerchio, è indistinguibile ad occhio nudo da un cerchio esatto; l'altro è descritto con spline di secondo grado e il massimo scostamento dall'altro cerchio ammonta solo a circa il 6% del raggio; ma messi a confronto la differenza è notevole.

Nota Nella lista dei riferimenti bibliografici tutti i manuali indicati fanno parte di ogni distribuzione completa del sistema TEX.

Riferimenti bibliografici

BECCARI, C. (2013). «Le filigrane e le figure di sfondo». *ArsTEXnica*, (15), pp. 46–58. URL http://www.guitex.org/home/numero-15.

— (2015). *The extension package curve2e*.

CARLISLE, D. (1999). *The trig package*. LATEX Project Team.

— (2014). *Packages in the 'graphics' bundle*. LATEX Project Team.

FEUERSÄNGER, C. (2015). *Manual for package PGFPLOTS – 2D/3D plots in LATEX*.

FUSTER, R. (2012). *The xpicture package – Several extensions of the picture standard environment*.

GÄSSLEIN, H., NIEPRASCHK, R. e TKADLEC, J. (2014). *The pict2e package*.

KNUTH, D. E. (1984). *The TEXbook*. Addison Wesley Publ. Co., Reading, MA.

MEHLICH, M. (1999). *The fp-package*. Il documento è un file di testo del manuale (UNIX-style).

OBERDIEK, H. (2009). *The picture package*.

TANTAU, T. (2013). *TikZ & PGF*.

THORUP, K.-K., JENSEN, F. e ROWLEY, C. (2014). *The calc package – Infix notation arithmetc in LATEX*. LATEX Project Team.

WICHURA, M. (1988). *PicTeX*. Il programma è libero, ma il manuale è a pagamento; un sommario abbastanza utile è il documento di Angus Duggan, *PiCTeX command summary*, 1990, disponibile in ogni distribuzione completa del sistema TEX.

▷ Claudio Beccari
claudio.beccari at gmail.com