

Inserire equazioni L^AT_EX in grafici di *Mathematica*

G. Gorni, S. Matiz

Sommario

Creare dei grafici da inserire nei propri articoli L^AT_EX o T_EX rappresenta certamente un problema soprattutto quando, per mantenere l'omogeneità dei simboli, si vogliono usare le font L^AT_EX anche all'interno del grafico. In questo articolo presentiamo una soluzione per coloro che usano e conoscono un po' la sintassi del software *Mathematica* e che conoscono L^AT_EX.

Con un unico comando, all'interno dell'interfaccia *Mathematica*

```
TeXClipping[sintassi LATEX, opzioni]
```

si ottiene un oggetto grafico **Graphics** per *Mathematica*: internamente è un insieme di poligoni e da un punto di vista visivo, non ha nulla da invidiare al suo collega font e si integra perfettamente nei grafici del suo ambiente.

Abstract

In this article we introduce a solution for creating graphics with the L^AT_EX fonts. This solution is meant for users that create pictures in *Mathematica* to be included in L^AT_EX documents. With a single command, within the *Mathematica* front end

```
TeXClipping[LATEX syntax, options]
```

we get a graphical object **Graphics** for *Mathematica*: a simple set of polygons that gives the same impression of its colleague the font and that integrates perfectly in the *Mathematica* ambient.

1 Introduzione

Il nostro proposito è stato quello di creare un sistema per gli utilizzatori di *Mathematica* che consentisse la creazione di grafici da inserire nei propri articoli L^AT_EX o T_EX, con le font di T_EX. Il sistema è di semplice utilizzo, non è necessario imparare nuovi linguaggi di programmazione e si integra perfettamente nell'ambiente di lavoro dando per esempio la possibilità in *Mathematica* versione 6 di copiare, incollare e spostare le formule L^AT_EX con il mouse all'interno di un grafico.

Sul web si possono trovare parecchie soluzioni per creare dei grafici con le font L^AT_EX da inserire nei propri articoli. Qui di seguito citiamo alcune tra le più diffuse

- PSTricks è un linguaggio di programmazione con sintassi affine a L^AT_EX che permette di creare grafici e disegni molto avanzati e usa naturalmente le font L^AT_EX.

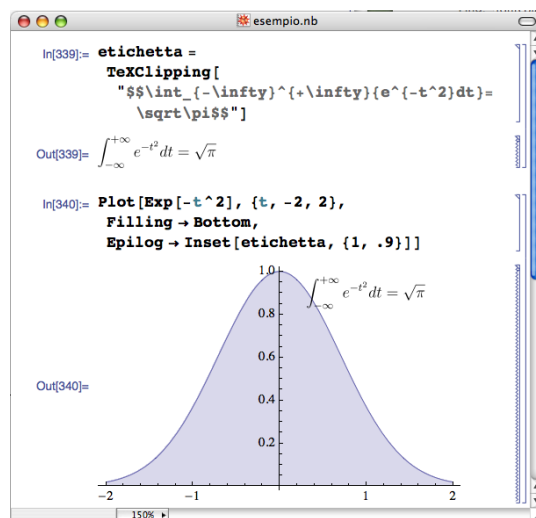


FIGURA 1: uno screenshot di *Mathematica* al lavoro

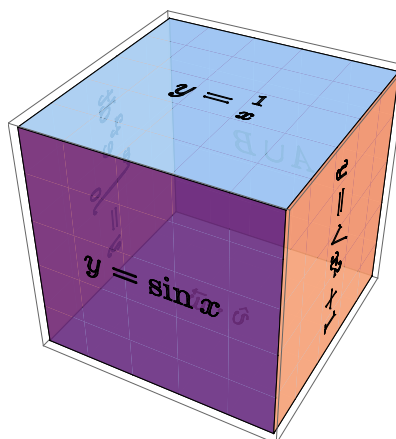


FIGURA 2: dado matematico

- WARMreader, di Ross Moore, permette di inserire delle etichette e delle annotazioni all'interno di figure e grafici importati in documenti T_EX e L^AT_EX.
- Gnuplot è un motore matematico che permette di trasformare il grafico prodotto in codice L^AT_EX pronto per essere compilato.
- METAPOST, Tikz e X_y-pic sono altri linguaggi di programmazione per il disegno simili a L^AT_EX ed in cui si possono usare le font L^AT_EX di default.

Il software commerciale *Mathematica* della Wolfram crea dei grafici piuttosto avanzati, e naturalmente il software si occupa anche di fare "i conti", ma *Mathematica*, ahimè, usa le proprie font che

$$\int_{-\infty}^{+\infty} e^{-t^2} dt = \sqrt{\pi}$$

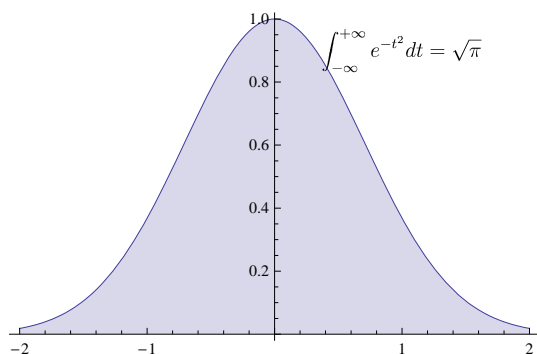
FIGURA 3: integrale in L^AT_EXiano visto da Mathematica

FIGURA 4: integrale e il suo grafico

sono diverse da quelle di T_EX. Il lavoro è stato quello di implementare un sistema per creare su Mathematica degli oggetti grafici che emulassero le font (e le formule) di L^AT_EX. Con Gnuplot è possibile fare una cosa molto simile, anche se in quel caso le formule L^AT_EX saranno vere font e non poligoni; inoltre sarà Gnuplot a creare il codice L^AT_EX del grafico e non viceversa. Il nostro approccio ha però il vantaggio che i poligoni della formula L^AT_EX si integrano nell'ambiente di Mathematica, quindi è possibile manipolarli come nell'esempio di figura 2 dove abbiamo inserito il grafico prodotto da TeXClipping in un grafico a tre dimensioni.

2 Installazione

È necessario aver installato i software gratuiti pstoedit e Ghostscript. A questo punto dopo aver caricato il pacchetto Mathematica dovrete già essere in grado di fare le prime prove a meno che un messaggio di errore vi avverta che, per qualche motivo, Mathematica non sia in grado di trovare gli eseguibili necessari (pstoedit o il compilatore T_EX). In questo caso è necessario che individuiate manualmente il percorso e settiate l'ambiente con il comando `SetOptions[PstoeditPath->..,TeXPath->..]`. Naturalmente una volta individuati gli eseguibili necessari questo comando iniziale sarà sempre lo stesso. Infine, precisiamo che gli esempi sono stati fatti con la versione 6 di Mathematica, ma che il pacchetto funziona anche nella precedente versione 5.

3 Uso di base

Un esempio di come il pacchetto può essere usato, in maniera semplice per l'utente, è il seguente (vedi figura 1):

- Carico il pacchetto, se non è caricato in automatico.
- Scrivo

```
etichetta=TeXClipping[
"$$\int_{-\infty}^{+\infty}
{e^{-t^2}}dt)=\sqrt{\pi}$$"
]
```

nel foglio di lavoro di Mathematica (notebook) e ottengo la figura 3.

- Scrivo

```
Plot[Exp[-t^2], {t, -2, 2},
Filling -> Bottom,
Epilog -> Inset[etichetta, {1, .9}]]
```

e ottengo la figura 4.

Tutto qui! Una sola funzione nuova: TeXClipping. Naturalmente ci sono alcune funzioni aggiuntive che permettono per esempio di scrivere le direttive precedenti con un unico comando TeXInset per ottenere direttamente la figura 4. Oppure alcune funzioni di debug: per esempio, se avessi dato il seguente comando: TeXClipping["\$\\beta"] (nel qual caso si produce, ovviamente, un errore) allora è possibile saperne di più con PrintError[], o cliccando sul tasto nel messaggio di errore, da cui otteniamo

Cannot compile the main.tex file!

```
\documentclass[12pt]{article}
\pagestyle{empty}\begin{document}
$\beta
\end{document}
:5:
Missing $ inserted
```

4 Le nuove funzioni

Le funzioni pubbliche sono le seguenti

- TeXClipping[comTeX,opzioni]: la funzione principale che trasforma la stringa comTeX di codice L^AT_EX in un grafico Mathematica. Con le opzioni FontColor e Magnification è possibile controllare il colore e le dimensioni della formula.
- TeXInset[comTeX,InsetArg]: funzione semplificatrice che, a partire dalla stringa comTeX, crea un grafico Mathematica `formLaTeX=TeXClipping[comTeX]` e restituisce un oggetto pronto per essere inserito in un grafico più grande.
- PrintError[]: Restituisce l'ultimo errore (utile soprattutto per fare il debug di una formula L^AT_EX).

- `GetFiles[dir]`: Per eventuali scopi di debug più approfonditi sposta i file ausiliari nella cartella *dir*.

5 Dettagli tecnici

Quello che abbiamo fatto è stato implementare una funzione di *Mathematica* che, prima con l'ausilio di pdfLATEX e poi con quello di `pstoedit`, trasformasse una stringa di codice TeX in un oggetto `Graphics` fatto di `Polygon`. Il primo passo è quello di individuare gli eseguibili necessari e creare una directory temporanea dove salvare i file ausiliari. Questo è ciò che fanno le seguenti funzioni

- `PrepareEnvironment`: la funzione cerca il compilatore LATEX, `pstoedit` e crea la cartella "TeXClipping" nella cartella temporanea di sistema.
- `GetPath[fileToFind]`: cerca il file *fileToFind*. Se il file è dato con il percorso completo restituisce semplicemente *fileToFind* altrimenti prima cerca il file di nome *fileToFind* nella cartella in cui si trova il nostro notebook `.nb` di lavoro corrente (il file sul quale stiamo lavorando); se non c'è, e solo in questo caso, lo cerca nella cartella del package `TeXClipping.nb` ed in entrambi i due casi ne restituisce il percorso completo. Se non lo trova affatto restituisce la stringa vuota.

I file ausiliari sono i seguenti

- `Main.tex` creato da *Mathematica* con all'interno il nostro codice LATEX.
- le funzioni usate sono `WriteTeXFileBody`, `WriteFile` e `WriteTeXFile`.
- `Main.pdf`, `Main.log`, `Main.aux` creati dal compilatore LATEX; e la funzione usata è `CompileTeX`.
- `pstoeditOutput.m` creato da `pstoedit`; e la funzione usata è `FormatPdf`.

Fin qui poche difficoltà, dato che finora il lavoro lo hanno fatto gli altri! Ora `pstoedit`, di Wolfgang Glunz, è un programma a linea di comando in grado di trasformare un file POSTSCRIPT o PDF in un'ampia gamma di altri formati vettoriali. È accessibile in maniera gratuita ed esporta in un oggetto `Graphics` di *Mathematica*. Precisamente, per i nostri scopi, `pstoedit` (con l'ausilio di `Ghostsript`) prende un file PDF creato da LATEX e ne restituisce un insieme di `Line`, linee che descrivono il contorno di ogni carattere. Purtroppo solo i contorni. Qui il nostro maggiore contributo: trasformare le linee in poligoni in modo che i caratteri risultassero anneriti nel loro interno. `pstoedit` restituisce un file di *Mathematica* pieno di linee e ora *Mathematica* deve interpretare le linee e formare i poligoni.



FIGURA 5: semplicemente "u" vista da *Mathematica*



FIGURA 6: la "o" sbagliata

5.1 Ottenere i poligoni

Trasformando ingenuamente un insieme di linee in un insieme di poligoni si può incorrere in qualche brutta sorpresa. Un semplice esempio chiarirà quale sia la problematica: durante l'esecuzione di `TeXClipping["u"]`, il `pstoedit` ci restituisce una linea che circonda la "u". Trasformando la linea in un poligono otterremo quello che vogliamo cioè una "u piena" come in figura 5. Ma se chiamiamo `TeXClipping["o"]`, da `pstoedit` otterremo due di linee, una per il cerchio più grande e una per quello più piccolo e trasformando tutto semplicemente in due poligoni otterremmo, visivamente, solo il cerchio grande annerito come in figura 6. Abbiamo peggiorato il risultato! Quello che dobbiamo fare è creare un unico poligono unendo i due cerchi. In questo modo la "o" si ottiene come una "u" dove avrei riunito le due estremità alte come in figura 7.

6 Dettagli ancora più tecnici (ma anche più interessanti)

Ci siamo imbattuti in alcuni problemi di topologia-geometria computazionale

6.1 La funzione `FillTeXSymbol`

La funzione `FillTeXSymbol` è la funzione *Mathematica* che si occupa di trasformare correttamente le linee ottenute da `pstoedit` in poligoni al fine di riempire i caratteri. Per prima cosa `FillTeXSymbol` trasforma l'insieme di liste di linee in un insieme di liste di poligoni X, come da manuale, poi applica ad X la regola `ruleForJoiningContainedPolygons` che individua le coppie di poligoni contenuti uno nell'altro e ne costruisce uno unico. Vediamo come avviene l'individuazione. Prendiamo una coppia di poligoni P_1 e P_2 . Se chiamiamo R_1 ed R_2 i più piccoli rettangoli contenenti tutti i punti dei rispettivi poligoni P_1 e P_2 , e risulta che R_1 è contenuto in R_2 , allora P_1 potrebbe essere contenuto in P_2 . In questo caso la funzione `rngMemberQ` restituirà vero se



FIGURA 7: semplicemente "o" vista da *Mathematica*

$$y = \sin x$$

FIGURA 8: risultato di `Import`

$$y = \sin x$$

FIGURA 9: risultato di `TeXClipping`

due poligoni hanno passato il primo test, e quindi sono candidati ad essere uno contenuto nell'altro; per sapere se effettivamente un poligono circonda l'altro, useremo la circuitazione. La funzione `circuitation` implementa un integrale numerico lungo la "curva" formata dai punti del poligono e restituisce un valore prossimo a zero se il punto `pt0` è circondato dalla curva `pts`.

7 Nota

La funzione `Import` di *Mathematica* versione 6 era stata la nostra prima scelta, ma purtroppo ha dei difetti che ci hanno costretti a indirizzarci a `pstoe-dit`. Immaginiamo di aver creato il file `main.pdf` compilando "`y = sin x`". Se noi vogliamo ottenere un risultato simile con `Import["main.pdf"]` otteniamo il risultato in figura 8 mentre con `TeXClipping` otteniamo il risultato in figura 9! Nel primo caso il "sin" è spostato a sinistra verso l'uguale e ci sono alcune linee sottili che però si vedono in stampa.

- ▷ G. Gorni
Gorni@dimi.uniud.it
- ▷ S. Matiz
matiz@dimi.uniud.it