Animazioni e grafici interattivi in LaTeX*

Grazia Messineo, Salvatore Vassallo

Sommario

In questo articolo illustriamo alcuni pacchetti LaTeX per la creazione di animazioni e grafici interattivi. Inoltre, mostriamo il loro utilizzo a livello didattico, in presentazioni, con il pacchetto beamer, e all'interno degli esercizi di autovalutazione on line prodotti con il pacchetto minerva.

Abstract

We present some LaTeX packages to create animations and interactive plots. We present also their use for educational applications, both in presentations, realized with the beamer package, and in the on line exercises created with the minerva package.

1 Introduzione

Quando si realizzano applicazioni didattiche (slides, esercizi di autovalutazione, ecc.) capita spesso che l'introduzione di animazioni o grafici interattivi faciliti la comprensione di un concetto. Si pensi, ad esempio, in ambito matematico, a quando si vuole illustrare come varia il grafico di una parabola al variare dei parametri.

In questo lavoro, passeremo rapidamente in rassegna alcuni strumenti esterni che consentono di realizzare animazioni in LaTeX, per poi introdurre alcuni pacchetti LaTeX che permettono di creare `pdf` animati o interattivi.

Infine, mostreremo alcune applicazioni, all'interno di presentazioni realizzate con il pacchetto beamer e all'interno di esercizi interattivi di autovalutazione, ottenuti con il pacchetto minerva (Messineo e Vassallo, 2013).

Nei prossimi paragrafi faremo alcuni esempi di ciò che è possibile realizzare con gli strumenti presentati. L'interattività dei file spesso rende del tutto inutile l'inserimento di immagini in un articolo. Per la visualizzazione degli esempi qui descritti rimandiamo alla presentazione realizzata per il convegno.

2 I programmi esterni

In questo paragrafo presentiamo brevemente alcuni programmi esterni, liberamente disponibili in rete, con i quali è possibile realizzare grafici interattivi da importare in LaTeX.

2.1 R

R è un software per la manipolazione di dati, il calcolo e la visualizzazione grafica. È orientato alla statistica.

La sua struttura è simile a quella di LaTeX: al centro del sistema vi è il nucleo, nel quale sono descritte le regole del linguaggio, che è modificabile solo dagli sviluppatori. Il nucleo è circondato da un primo anello nel quale è contenuta una serie di funzionalità di base (gli algoritmi che consentono di costruire le funzioni statistiche). Più all'esterno sono contenuti tutti i pacchetti che coprono molte esigenze di tipo statistico.

R può essere usato insieme a LaTeX mediante l'uso di alcuni pacchetti specifici, dei quali il più conosciuto è Sweave, presentato nel convegno GUIT 2014 (Baltieri e Scandola, 2014).

Le animazioni in LaTeX possono essere create in due modi:

- inserendo direttamente il codice R nel file LaTeX e usando per la compilazione il pacchetto Sweave e per le animazioni il pacchetto animate (v. par. 3.1.1);
- creando in R dei file di immagine che poi vengono importati nel file LaTeX e animati sempre con l'uso del pacchetto animate.

2.2 Asymptote

Asymptote è un linguaggio grafico vettoriale molto potente che mette a disposizione dell'utente un ambiente basato su coordinate per il disegno tecnico. Le etichette e le equazioni sono scritte in LaTeX, in modo da generare un output PostScript di alta qualità.

Asymptote può essere usato direttamente in LaTeX attraverso il pacchetto omonimo, che mette a disposizione due ambienti fondamentali:

- asydef, che contiene le definizioni globali del documento;
- asy, che contiene il codice Asymptote vero e proprio.

Il file `pdf` è ottenuto con i seguenti comandi:

```
pdflatex filename
asy filename-1.asy
pdflatex filename
```

*Il lavoro si inserisce nel progetto di ricerca di interesse d'Ateneo dell'Università Cattolica "Progetto M.In.E.R.Va.: Creazione di esercizi di tipo interattivo con percorsi differenziati per il recupero delle carenze e la valorizzazione delle eccellenze in matematica".

I grafici che si ottengono possono essere sia statici che interattivi (l'utente può ruotarli, ecc.).

La realizzazione di grafici ottenibili programmando Asymptote è stata illustrata durante il convegno GUIT 2009. (De Marco, 2009)

2.3 Sage

Sage è un programma multipiattaforma, scritto in Python, orientato alla matematica, che consente di risolvere problemi di varia natura (analisi, calcolo combinatorio, ecc.).

Il programma è disponibile per Linux e per Mac Os X; per Windows è stata implementata una versione che può essere utilizzata insieme alla Oracle VirtualBox. Esiste anche la possibilità di utilizzarlo (anche insieme a LaTeX) in cloud (`https://cloud.sagemath.com/`).

Sage può essere utilizzato con LaTeX in vari modi.

Ad esempio, il pacchetto sagetex (Drake *et al.*, 2009) consente di usare codice sage nel documento e fa sì che questo codice sia correttamente compilato durante la compilazione LaTeX. Il codice può essere interpretato sia con un'installazione locale di `sage`, sia utilizzando un'installazione su server remoto, attraverso uno script Python.

Esiste anche il programma sws2tex (Huss e Maříк, 2009), un convertitore in grado di trasformare i file `sws` di sage in file `tex` e di compilarli, producendo in output un file `pdf`.

Sage consente di realizzare animazioni e grafici interattivi di vario tipo, che possono essere salvati in `pdf` e utilizzati all'interno di file LaTeX.

3 I pacchetti LaTeX per le animazioni

In questo paragrafo presentiamo le strategie che si possono usare in LaTeX per la realizzazione di grafici animati e alcuni pacchetti che consentono di ottenerli. Tali pacchetti permettono di ottenere grafici interattivi con o senza l'ausilio di programmi esterni.

Tutti i grafici animati realizzati con questi pacchetti vengono visualizzati correttamente con Adobe Acrobat Reader e, in alcuni casi, con altri visualizzatori (ad esempio, PDF-XChange in ambiente Windows).

3.1 Le strategie per ottenere grafici animati

In LaTeX è possibile ottenere grafici animati usando due strategie:

- realizzare, mediante codice LaTeX o programmi esterni, delle immagini in formati compatibili, da inserire nel file principale attraverso l'uso di appositi pacchetti: animate (v. par. 3.1.1) per tutti i file di immagine compatibili con il motore usato per la compilazione; movie15 se si vogliono inserire filmati o, in particolare, file di immagine in formato `gif`, come le `gif` animate (v. par. 3.1.2).
- inserire direttamente il codice che genera l'immagine all'interno di un ambiente in grado di generare contemporaneamente le immagini e la loro animazione: tale obiettivo viene raggiunto con il pacchetto animate.

Nei prossimi paragrafi presenteremo i due pacchetti utili per la realizzazione di animazioni.

3.1.1 Animate

Il pacchetto animate di Alexander Grahn (Grahn, 2015a) consente di creare file `pdf` con contenuti animati, ottenuti da un insieme di file grafici o file immagine, così come da grafici generati con PSTricks o Ti*k*Z, oppure semplicemente dal testo. Permette di creare animazioni anche con grafici vettoriali.

I formati immagine accettati sono tutti quelli compatibili con il motore scelto per la compilazione: `eps`, `ps` e `mps` se si usa LaTeX, dvips e ps2pdf; `pdf`, `mps`, `png`, `jpg`, ecc. se si usa pdfLaTeX.

Il pacchetto mette a disposizione due strumenti fondamentali.

- Il comando `\animategraphics`, che consente di creare animazioni da insiemi di file grafici creati esternamente oppure da file `pdf` che contengono più pagine. La sintassi del comando è

```
\animategraphics[<opzioni>]
{<frame rate>}
{<nome di base del file>}
{<primo>}{<ultimo>}
```

  Tra le opzioni: può essere indicata la *timeline*, scritta su un file esterno che viene successivamente caricato; può essere assegnata un'etichetta (univoca); può essere impostata una pausa automatica, ecc.

  Il parametro frame rate indica il numero di fotogrammi che vengono visualizzati in un secondo; nome di base del file indica la parte comune dei nomi assegnati ai file di immagine da usare per l'animazione (ad esempio, se i file di immagine sono `file01.eps`, `file02.eps`, ecc., nome di base del file è `file`); primo e ultimo sono, rispettivamente, il primo e l'ultimo numero assegnato ai file di immagine.

- L'ambiente animateinline, che consente di creare animazioni dal codice racchiuso al suo interno. Il codice all'interno può essere creato mediante l'ambiente standard di LaTeX, picture, oppure usando i pacchetti pstricks (par. 3.2.2) o tikz/pgf (par. 3.2.3).

  La sintassi dell'ambiente è

```
\begin{animateinline}
[<opzioni>]{<frame rate>}
\dots codice ...
\newframe[<frame rate>]
\dots codice ...
\newframe*[<frame rate>]
\dots codice ...
\newframe
\multiframe{<numero di fotogrammi>}
{[<variabili>]}
{...codice ripetuto...}
\end{animateinline}
```

Le opzioni dell'ambiente sono simili a quelle del comando `\animategraphics`, così come il significato del parametro frame rate.

Il comando `\newframe` serve per chiudere il frame precedente ed iniziarne uno nuovo; la versione con * prevede una pausa dopo il frame dietro al quale è posizionato. L'animazione riprende cliccando con il mouse.

Il comando `\multiframe` consente di costruire dei loop con le figure. Per creare il loop, si usa il parametro opzionale variabili, che contiene una lista di variabili (separate da virgola) che possono essere inizializzate ad un determinato valore e incrementate di un valore predefinito e fisso.

Si rinvia alla documentazione del pacchetto (Grahn, 2015a) per ulteriori dettagli sulle opzioni del comando e dell'ambiente e anche per alcuni esempi di utilizzo.

3.1.2 Movie15

Il pacchetto movie15 di Alexander Grahn (Grahn, 2012) consente di inserire all'interno di un file LaTeX immagini in formato `gif`, in particolare `gif` animate.

Il pacchetto è, per la verità, obsoleto e lo stesso autore consiglia l'uso del pacchetto più recente media9 (Grahn, 2015b), che però non consente di inserire direttamente `gif` animate.

Il pacchetto mette a disposizione, per il caricamento del file, il seguente comando:

```
\includemovie[<opzioni>]{<larghezza>}
{<altezza>}{<file>}
```

Nelle opzioni è possibile specificare se la riproduzione e l'arresto del file devono essere automatici, se si vuole che il file sia aperto in un visualizzatore esterno, ecc.

Se non si desidera o non si può usare movie15, è necessario convertire la gif animata in uno dei formati supportati da animate o da media9 e usare uno dei due pacchetti.

3.2 I pacchetti per la realizzazione di immagini

In questo paragrafo presentiamo tre pacchetti: il pacchetto multido, che consente di automatizzare alcune operazioni ripetute e due pacchetti per la realizzazione di immagini, pstricks e tikz/pgf.

Non ci addentreremo in questa sede in spiegazioni dettagliate sul funzionamento dei pacchetti, ci limiteremo ad illustrare come possono essere adoperati per la creazione di immagini da rendere "animate".

3.2.1 Il pacchetto multido

Il pacchetto multido (Van Zandt, 2010) di T. Van Zandt, creato inizialmente per il pacchetto pstricks, ma utilizzabile anche indipendentemente da esso, mette a disposizione il comando omonimo, che consente di automatizzare una serie di operazioni ripetitive mediante un'unica istruzione.

La sua sintassi è:

```
\multido{variabili}{ripetizioni}{codice}
```

Il primo parametro obbligatorio, variabili, consente di dichiarare le variabili (separate da virgola) da usare nel disegno. Devono essere definite nella forma `variabile=valore iniziale+incremento` e l'incremento deve essere preceduto dal segno − se si desidera decrementare la variabile.

Il parametro ripetizioni indica il numero di volte in cui si vuole ripetere un'azione. Il parametro codice contiene invece il codice che si vuole ripetere.

Tale comando risulta particolarmente utile, tra le altre cose, per la produzione di file di immagine da rendere animati.

3.2.2 PSTricks

PSTricks è una collezione di macro di interfaccia tra TeX (e LaTeX) e il linguaggio PostScript.

Consente di disegnare (quasi) tutti i tipi di oggetti, grafici 2D e 3D, a colori e non, e molto altro.

Usando PSTricks, un qualsiasi disegno viene "tracciato" attraverso una serie di comandi.

Poiché PSTricks è un insieme di macro in linguaggio PostScript, per visualizzare l'output correttamente si hanno due opzioni:

- compilarlo con LaTeX, `dvips`, `ps2pdf`;
- usare il pacchetto auto-pst-pdf oppure passare l'opzione pdf al pacchetto (che chiama auto-pst-pdf). Ciò consente la compilazione diretta dei file contenenti codice PSTricks con pdfLaTeX.

Il pacchetto PSTricks è molto ben documentato: in rete sono presenti diversi manuali e tutorial, ai quali si rimanda per una descrizione delle sue funzionalità di base e avanzate.

PSTricks consente di realizzare animazioni in due modi.

- Creando dal codice PSTricks file `pdf` con immagini interattive (dalle quali si possono poi ottenere `gif` animate, da usare in siti web o in altri programmi): in questo caso, si può usare il comando `\multido` già descritto nel par. 3.2.1.

Si consideri il seguente (semplicissimo) esempio:

```
% nel preambolo
\usepackage{multido,fp}
\usepackage{pstricks-add}
...
\multido{\n=0+1}{20}{
\begin{pspicture}(-3,-3)(3,3)
\ifodd\n
\FPeval\r{round((\n/10):1)}
\pscircle[linecolor=yellow]{\r}
\else
\FPeval\r{round((\n/10):1)}
\pscircle{\r}
\fi
\end{pspicture}
}
```

Questo codice produce una serie di immagini composte da cerchi di dimensione crescente colorati alternativamente di giallo e di nero. Per ottenere cerchi di dimensione decrescente, è sufficiente definire la variabile `\n` a partire dal valore massimo e decrementarla anziché incrementarla.

Una volta creato il file `pdf`, è possibile inserirlo direttamente in un file LaTeX usando il comando `\animategraphics` del pacchetto animate (3.1.1). In alternativa, è possibile ottenere una `gif` animata (da inserire ad esempio in una pagina `html` o in presentazioni realizzate non in LaTeX) usando ImageMagick e il comando

```
convert -delay x -loop 0
-density y -scale z
-alpha remove
file.pdf file.gif
```

La `gif` animata così ottenuta può essere utilizzata in un file LaTeX con l'ausilio del pacchetto movie15.

- È possibile anche inserire il codice per creare le immagini all'interno dell'ambiente animateinline (par. 3.1.1), come mostrato nel seguente esempio tratto da https://www.tug.org/PSTricks/main.cgi?file=Animation/basics:

```
\newwrite\TimeLineFile
\immediate\openout
\TimeLineFile=sinus.txt
\immediate\write
\TimeLineFile{::0x0,1}%
\multido{\i=2+1}{90}{%
\immediate\write\TimeLineFile{%
::\i}
}
\immediate\closeout\TimeLineFile
```

Queste istruzioni servono per scrivere automaticamente il contenuto del file della timeline.

```
\psset{xunit=\pstRadUnit,
dashadjust=false}
\begin{animateinline}
[controls,timeline=sinus.txt,
begin={\begin{pspicture}
(-0.5,-1.5)(6.6,2)},
end={\end{pspicture}}]{10}
\psaxes[trigLabels,trigLabelBase↩
    =3]{->}(0,0)(-.2,-1.5)(6.5,1.5)
[$t$,-90][$y=\sin(t)$,0]
\psplot[linestyle=dashed,xunit=1cm,
algebraic]{0}{\psPiTwo}{sin(x)}
\newframe
\multiframe{91}{n=0+4}{
\psset{xunit=1cm,linecolor=blue}
\pscustom[xunit=1cm,fillcolor=blue,
fillstyle=solid,linestyle=none,
algebraic,dimen=inner]{%
\psplot{0}{\n\space DegtoRad}
{sin(x)}
\psline(!\n\space DegtoRad 0)}
\psplot[linestyle=dashed,xunit=1cm,
linecolor=black,algebraic]{0}
{\n\space DegtoRad}{sin(x)}
\psdot[opacity=0.4,dotsize=3mm]
(!\n\space dup DegtoRad exch sin)
\psline[linestyle=dashed]
(!\n\space dup DegtoRad exch sin)
(!\n\space DegtoRad 0)}
\end{animateinline}
```

Con queste istruzioni si crea l'animazione inserendo il codice per l'immagine direttamente all'interno dell'ambiente e si inseriscono i pulsanti che permettono di controllarla. Il grafico che ne risulta è quello della funzione $y = \sin x$, con un punto che si muove su di esso e l'area compresa tra il grafico della funzione, l'asse x e la retta $x = x_0$, con x_0 ascissa del punto mobile, che si colora automaticamente.

3.2.3 TikZ/Pgf

Il pacchetto pgf e la sua interfaccia (non grafica) Ti*k*Z costituiscono un altro approccio alla grafica in LaTeX. Infatti permettono di costruire figure mediante comandi simili a quelli di altri programmi o pacchetti, e di avere come output sia un file PostScript, sia un file `pdf`, senza l'intervento di programmi esterni, ma con l'usuale compilazione con LaTeX o pdfLaTeX. Il fatto di essere compatibile con pdfLaTeX ha però l'inconveniente di una minore potenza del pacchetto rispetto a PSTricks in quanto il linguaggio PDF è meno potente (soprattutto da un punto di vista matematico) del linguaggio PostScript.

L'autore dei pacchetti è Till Tantau (autore anche del pacchetto beamer). La loro documentazione (Tantau, 2013) è molto ricca.

Per usare il pacchetto tikz è necessario caricarlo direttamente, oppure caricare il pacchetto pgf. Per particolari scopi, si possono caricare altre librerie specializzate di pgf, ad esempio per disegnare diversi tipi di frecce o per usare sfondi, ecc.

Presenteremo qui solo alcuni esempi di utilizzo del pacchetto pgf per produrre grafici animati. L'idea di base è la stessa già illustrata per il pacchetto pstricks (v. 3.2.2).

- La prima possibilità è quella di creare file di immagine in formato pdf, o altro formato riconosciuto da LaTeX e successivamente utilizzare il comando \animategraphics del pacchetto animate per creare l'animazione.

  Ad un risultato simile si può pervenire usando, insieme a TikZ, il pacchetto tkz-fct, che si basa sul programma di grafica Gnuplot (esempio tratto da http://tex.stackexchange.com/questions/106140/definite-integral-animation-examples):

```
% nel preambolo
\usepackage{tkz-fct}
\usepackage{multido}
...
\multido{\i=2+1}{20}{%
\begin{tikzpicture}[scale=1.25]
\tkzInit[xmax=8,ymax=4]
\tkzAxeXY[ticks=false]
\tkzGrid
\tkzFct[color=red,
    domain=0.125:8]{4./x}
\tkzDrawRiemannSumInf[fill=green,
opacity=.2,color=green,interval=1:8,
line width=1pt,number=\i]
\end{tikzpicture}}
```

  Le immagini ottenute con questo documento possono poi essere animate usando il comando \animategraphics.

  Per ottenere le immagini da inserire nel comando \animategraphics è possibile utilizzare il pacchetto pgfplots, che illustreremo più dettagliatamente nel paragrafo 3.3.2. Tale uso è mostrato in questo esempio:

```
\foreach \pos in {0,0.05,...,1.05}{
\begin{tikzpicture}
\begin{axis}[grid= major,xlabel=$x$,
  ylabel =$y$,
  ylabel style={rotate=-90},
  no marks ]
 \addplot{\x^2}
 node[fill=orange,draw=blue,circle,
 inner sep=1pt, pos=\pos]{};
\end{axis}
\end{tikzpicture}}
```

- La seconda possibilità è creare l'immagine all'interno dell'ambiente animateinline, come viene fatto in questo esempio tratto da http://www.texample.net/tikz/examples/animated-definite-integral/:

```
\newcounter{angle}
\setcounter{angle}{0}
\newcounter{r}
\newcommand{\escalar}[1]{
\setcounter{r}{#1 * #1 * #1}}
\newcounter{m}
\setcounter{m}{0}
\newcounter{mc}
\begin{animateinline}[loop,controls,
 poster= first,palindrome]{2}
\whiledo{\them < 21}{
\begin{tikzpicture}[scale=1.25]
\draw[red,thick,<->] (-1,1)
parabola bend (0,0) (2.1,4.41)
node[below right] {$y=x^2$};
\draw[loosely dotted] (-1,0)
grid (4,4);
\draw[->] (-0.2,0) -- (4.25,0)
node[right] {$x$};
\draw[->] (0,-0.25) -- (0,4.25)
node[above] {$y$};
\foreach \x/\xtext in {1/1, 2/2, ↩
    3/3}
\draw[shift={(\x,0)}] (0pt,2pt) --
(0pt,-2pt) node[below] {$\xtext$};
\foreach \y/\ytext in
{1/1, 2/2, 3/3, 4/4}
\draw[shift={(0,\y)}] (2pt,0pt) --
(-2pt,0pt) node[left] {$\ytext$};
\setcounter{mc}{\value{m}*\value{m}}
\shade[top color=blue,
bottom color=gray!50](0,0)
parabola(0.1*\them,0.01*\themc)
|- (0,0);
\escalar{\them}
\draw (3cm,2pt)node[above]
{$\displaystyle\int\0^{\them/10}
\!\!x^2\mathrm{d}x =
\displaystyle\frac{\ther}{3000}$};
\draw[fill=orange,color=orange]
(0.1*\them,0.01*\themc)circle(.5pt);
\end{tikzpicture}
\stepcounter{m}
\ifthenelse{\them < 21}{
\newframe
}
{\end{animateinline}\relax}
}
```

3.3 I grafici interattivi

In questo paragrafo presentiamo due pacchetti per la creazione di grafici interattivi: interactiveplot e pgfplots.

Definiamo grafici interattivi quei grafici dei quali l'utente può modificare in vari modi il comportamento (ad esempio variando, mediante pulsanti, il valore di uno o più parametri) o che, cliccando con il mouse, mostrano informazioni di vario tipo relative all'oggetto disegnato (ad esempio, le coordinate di un punto).

3.3.1 Interactiveplot

Interactiveplot è un pacchetto realizzato da R. Bock, P. Linares e J. Toro (Bock *et al.*, 2014) che consente di creare, all'interno di un file pdf, grafici parametrici in due e tre dimensioni. Il comportamento del grafico al variare dei parametri può essere controllato da bottoni che li fanno aumentare o diminuire secondo un passo prefissato.

Un grafico 2D viene inserito nell'ambiente iplotdd, un grafico 3D nell'ambiente iplotddd.

All'interno dei due ambienti, l'espressione algebrica viene inserita come argomento del comando \iplot e l'interattività gestita attraverso alcune delle sue opzioni.

Per i grafici 2D, la sintassi è

```
\begin{iplotdd}[width=w,height=h]
\iplot[var={nome_v,vmin,vmax},
```

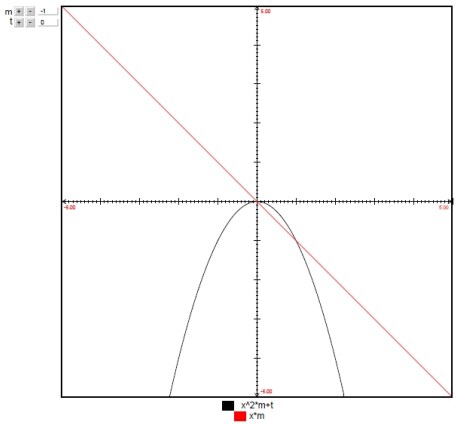

FIGURA 1: Una parabola e una retta variabili

```
param={nome_p,valore_iniz,step,min,max},
frange={nome,min,max},color=colore]
{funzione}
\end{iplotdd}
```

L'opzione var consente di definire la variabile indipendente e i suoi valori minimo e massimo. Il grafico sarà disegnato nell'intervallo $(vmin, vmax)$. L'interattività è prodotta dall'opzione param, mediante la quale vengono definiti uno o più parametri che compaiono nell'espressione algebrica della funzione. Ogni parametro è definito da un nome (nome_p). Deve essere attribuito un valore iniziale (il primo che compare sul grafico e per il quale esso è disegnato), il passo con cui il parametro deve essere diminuito o aumentato, il valore minimo e massimo. Se non vengono definiti parametri, il grafico diventa statico. Per ogni parametro, vengono inseriti nel file pdf due bottoni, uno con un segno + e l'altro con un segno −, che consentono di aumentare o diminuire il valore del parametro di step unità. L'opzione frange consente di definire la variabile dipendente e il suo intervallo di variazione.

La sintassi per i grafici 3D è identica, con l'unica differenza che devono essere definite due variabili indipendenti (sempre con l'opzione var).

Nello stesso ambiente possono essere inseriti più grafici dello stesso tipo (2D o 3D). È anche possibile usare lo stesso parametro per due o più grafici: in questo caso, la modifica del valore del parametro mediante i pulsanti + e − modifica l'aspetto di tutti i grafici che contengono quel parametro. Nell'esempio che segue, lo stesso parametro viene usato sia nel primo che nel secondo grafico (si veda la figura 1):

```
\begin{iplotdd}[width=400,height=400]
\iplot[var={x,-5,5},
param={m,-1,1,-10,10},
param={t,0,1,-10,10},
color=black,
frange={y,-5,5}]
{x^2*m+t}
\iplot[var={x,-5,5},
param={m,-1,1,-10,10},
color=red,
frange={y,-5,5}]
{x*m}
\end{iplotdd}
```

L'interattività dei grafici è garantita dall'inserimento di codice Javascript nel file pdf.

Il pacchetto ha buone potenzialità, ma al momento "soffre" di alcune limitazioni:

- non è possibile controllare la posizione dei pulsanti che animano il grafico;
- non è possibile controllare la posizione e lo stile della legenda dei grafici disegnati. Inoltre, i parametri sono sempre indicati con il loro nome e non vengono sostituiti i valori man mano loro attribuiti;
- non è possibile inserire punti, etichette, ecc.
- il pacchetto non funziona con le principali classi utilizzate per produrre presentazioni (beamer, powerdot, ecc.).

3.3.2 Pgfplots

Il pacchetto pgfplots, basato sul pacchetto pgf descritto nel paragrafo 3.2.3, è stato scritto da Christian Feuersänger, che ne cura anche la documentazione (FEUERSÄNGER, 2015).

Una descrizione di base molto dettagliata delle funzionalità del pacchetto è stata presentata al convegno GUIT del 2011 (DE MARCO e GIACOMELLI, 2011).

Anche in questo caso presentiamo solo alcune applicazioni dei comandi del pacchetto alla produzione di grafici interattivi, rimandando al manuale e all'articolo per ogni altra esigenza.

Presentiamo due applicazioni realizzate con pgfplots.

- La prima è un grafico che mostra come varia il comportamento della parabola $y = ax^2$ al variare del parametro a:

```
% nel preambolo
\usepackage{pgfplots}
\usetikzlibrary{ocgx}
\tikzset{ocg button/.style={circle,
inner sep=.25em,switch ocg
with mark on={#1}{}}}
\tikzset{base/.style=
{baseline=-0.5ex}}
\newcommand{\function}{\x^2}
\newcommand{\button}[2]{\tikz[base]
\node[fill=#2!30,ocg button=#1]{};}
\newcommand{\plotit}[2]
{\addplot+[ocg={name=#2,ref=#2}]
{#1};\label{#2}}
\newcommand{\legendit}[3]
{\item[\ref{#1}]
#2$x^2$ \button{#1}{#3}}
...
\begin{minipage}[b][0.4\textheight]
[l]{0.7\textwidth}
\begin{tikzpicture}
```

```
\begin{axis}[grid= major,xlabel=$x$,
ylabel =$y$,
ylabel style={rotate=-90},
cycle list={blue,red,green!50!lime,
orange,cyan!50!blue},]
 \plotit{\function}{first}
 \plotit{1.5*\function}{second}
 \plotit{2.25*\function}{third}
 \plotit{3*\function}{fourth}
 \plotit{5*\function}{fifth}
\end{axis}
\end{tikzpicture}
\end{minipage}
\begin{minipage}[b][0.5\textheight]
[c]{0.2\textwidth}
\begin{itemize}
\legendit{first}{}{blue}
\legendit{second}{1.5}{red}
\legendit{third}{2.25}{green}
\legendit{fourth}{3}{orange}
\legendit{fifth}{5}{cyan}
\end{itemize}
\end{minipage}
```

L'esempio presentato fa uso della libreria ocgx, che il pacchetto ocgx (Isambert e Gaborit, 2012) mette a disposizione per l'uso diretto con Ti*k*Z e consente di rendere visibili o invisibili gli oggetti, creando un bottone da cliccare per farli apparire o scomparire.

- La seconda applicazione consente di avere grafici con coordinate cliccabili. Questo "effetto" è ottenuto mediante l'uso della libreria `clickable` di pgfplots, come mostra il seguente semplice esempio:

```
% nel preambolo
\usepackage{pgfplots}
\pgfplotsset{compat=newest}
\usepgfplotslibrary{clickable}
...
\pgfplotsset{/pgfplots/annot/%
point format={(\%.2f, \%.2f)}}
\begin{tikzpicture}
\begin{axis}[]
\addplot[red,domain=-3:3,
samples=201,]{x^2};
\end{axis}
\end{tikzpicture}
```

Questo codice consente di ottenere il grafico di una parabola con coordinate cliccabili: aprendo il file pdf con Adobe Acrobat Reader e cliccando su un qualsiasi punto del grafico, appaiono le coordinate del punto cliccato. È ovviamente possibile personalizzare il comportamento delle finestre pop up che appaiono cliccando e altri aspetti (si veda la figura 2).

Un risultato simile può essere ottenuto inserendo i punti da coordinate date in tabella, come mostra questo esempio, tratto dal manuale di pgfplots (Feuersänger, 2015):

```
% preambolo
\usepackage{pgfplots}
\pgfplotsset{compat=newest}
\pagestyle{empty}
\usepgfplotslibrary{clickable}
```

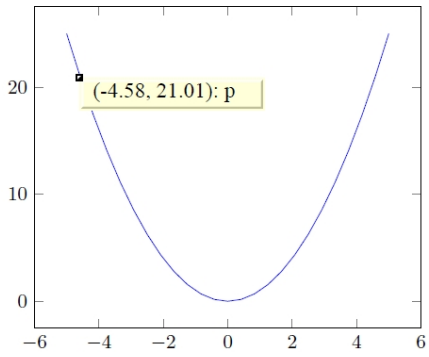

Figura 2: Una parabola con coordinate cliccabili

```
...
\begin{tikzpicture}
\begin{axis}[%
clickable coords={(xy):
\thisrow{label}},%
scatter/classes={%
a={mark=square*,blue},%
b={mark=triangle*,red},%
c={mark=o,draw=black}}]
\addplot[scatter,only marks,%
scatter src=explicit symbolic]%
table[meta=label] {
x      y       label
0.1    0.15    a
0.45   0.27    c
0.02   0.17    a
0.06   0.1     a
0.9    0.5     b
0.5    0.3     c
0.85   0.52    b
0.12   0.05    a
0.73   0.45    b
0.53   0.25    c
0.76   0.5     b
0.55   0.32    c
};
\end{axis}
\end{tikzpicture}
```

Questo codice produce un grafico con punti creati dalle coordinate x e y delle prime due colonne. Il simbolo del punto (cerchio, quadrato, triangolo) è attribuito in base alla lettera che compare nella terza colonna. Le coordinate del grafico sono cliccabili: vengono mostrate sia le coordinate dei punti, che la lettera che consente di attribuire il simbolo.

La libreria `clickable` rende cliccabile l'intera area del grafico e mostra con finestre pop-up le coordinate di ogni punto di quell'area, indipendentemente dal fatto che sia un punto del grafico oppure no. Nella sezione riguardante le applicazioni didattiche (v. par. 4) presenteremo alcune possibili strategie di risoluzione di questo "inconveniente".

4 Applicazioni didattiche

I pacchetti finora presentati hanno applicazioni molto utili dal punto di vista didattico: spesso gli studenti faticano a "vedere" ciò che si spiega loro e le animazioni, i grafici interattivi e quelli cliccabili possono aiutare a risolvere questi problemi.

Presentiamo qui due applicazioni: le lezioni teoriche (servendoci di pacchetti per le presentazioni, ad esempio beamer) e le esercitazioni on line realizzate con il pacchetto minerva (Messineo e Vassallo, 2013).

4.1 Le lezioni teoriche

In genere, durante una lezione teorica, il supporto delle slide diventa fondamentale quando si vuole illustrare graficamente un certo concetto o una certa proprietà.

Alcuni esempi molto efficaci didatticamente possono ad esempio essere trovati sul sito del prof. Mařík (http://user.mendelu.cz/marik/latex/).

Si pensi, ad esempio, di voler illustrare come varia il grafico della funzione $y = (x \pm p)^2$ al variare del parametro p. Presentiamo due soluzioni.

- La prima è realizzata con i pacchetti beamer, pgfplots e animate e prevede la realizzazione di un file pdf di immagini composto da varie pagine (con l'uso del comando \foreach di pgfplots, che dà risultati simili al comando \multido già illustrato), che viene poi "animato" mediante l'uso del comando \animategraphics.

  Il seguente codice consente di creare le immagini

```
  % nel preambolo
\usepackage{pgfplots}
...
\foreach\p/\y in{0/{white,opacity%
=0},5/green,10/red,15/brown,%
20/orange,25/yellow}
{\begin{tikzpicture}
\begin{axis}[xlabel=$x$,
ylabel =$y$,
ylabel style={rotate=-90},
no marks,
axis x line=center,
axis y line=center,
yticklabels={}]
\addplot[color=black,line width=2pt,
domain=-22:22,opacity=0.2, dashed]%
{x^2}node[right,pos=.1]{$ x^2$};
\pgfmathparse{\p+22}%
\let\a=\pgfmathresult
\pgfmathparse{\p-22}%
\let\b=\pgfmathresult
\edef\temp{
\noexpand\addplot[color=\y,
line width=2pt,domain=\b:\a]
{(x-\p)^2} node[right,pos=.1]
{\noexpand\scriptsize{$(x-\p)^2$}}
node[coordinate,pin=above:%
{\noexpand\scriptsize{$V(\p,0)$}}]%
at (axis cs: \p,0){} ;}
\temp\end{axis}
\end{tikzpicture}}
```

  L'uso del comando \foreach consente in questo caso di utilizzare un ciclo nel quale ruotano contemporaneamente due variabili: la prima è il parametro p, che in questo caso assume solo i valori di una lista predefinita, la seconda è il colore che il grafico deve assumere in corrispondenza di ogni valore di p.

  L'animazione viene prodotta invece con il seguente codice:

```
% preambolo
\usepackage{pgfplots}
\usepackage{animate}
...
\begin{frame}[fragile]%
{Grafico di $y=(x-p)^2$}
\animategraphics[label=parabola,
controls]{1}{parabola}{}{}
\end{frame}
```

- La seconda è, invece, realizzata con l'uso del pacchetto interactiveplot che, come già detto, al momento non può essere utilizzato insieme ai principali pacchetti per le presentazioni (v. par. 3.3.1):

```
% nel preambolo
\usepackage{interactiveplot}
...
\begin{iplotdd}[width=400,
height=400]
\iplot[var={x,-5,5},color=red,
 param={p,0,1,-5,5}]  {(x-p)^2}
\end{iplotdd}
```

4.2 Le esercitazioni online

Presso alcune scuole superiori delle provincie di Milano e Varese e presso l'Università Cattolica del Sacro Cuore è in atto da quattro anni la sperimentazione di una piattaforma on line di esercizi di autovalutazione di Matematica (Generale e Finanziaria): si tratta della piattaforma "M.In.E.R.Va." già illustrata durante il convegno GUIt 2012 (Messineo e Vassallo, 2012). Gli studenti dell'Università Cattolica coinvolti nella sperimentazione svolgono online anche una parte della prova d'esame.

Per la realizzazione degli esercizi di autovalutazione, abbiamo realizzato il pacchetto minerva, illustrato durante il Convegno GUIt 2013 (Messineo e Vassallo, 2013).

Presentiamo qui un esempio di come i grafici interattivi possano essere utilizzati per rendere maggiormente comprensibile la modalità di svolgimento di un esercizio.

Si supponga che sia assegnato un grafico e che si chieda di individuare una possibile espressione algebrica per la funzione. Nel file che contiene tutti gli esercizi relativi all'argomento in questione, sarà presente un problema di questo tipo:

```
\newproblem{
\FPsetpar{a}{2}{6}
\FPsetpar{b}{1}{\a}[\a]
\item \PTs{1} Sia $f(x)$ la funzione
il cui grafico è
\begin{center}
\begin{tikzpicture}
\begin{axis}[xlabel=$x$,  ylabel =$y$,
  ylabel style={rotate=-90},
   axis x line=center,
axis y line=center,ymin=0,]
  \addplot[color=green!50!lime,%
  domain=-1.5:1.5]{\b+\a*x^2};
\end{axis}
\end{tikzpicture}
\end{center}
una possibile espressione algebrica
di $f(x)$ è
\begin{answers}*{1}
\bChoices[random]
\Ans0 $f(x)=\b+x^2$  \eAns
\Ans0 $f(x)=\a x^2$ \eAns
\Ans1 $f(x)=\b+\a x^2$  \eAns
\Ans0 $f(x)=\b-\a x^2$  \eAns
\Ans0 nessuna delle altre risposte è
    corretta \eAns
\eChoices
\end{answers}
\begin{solution}
Si consideri il seguente grafico.
Si parte dalla funzione elementare
$y=x^2$ (in blu), si traccia poi il
grafico di $y=\a x^2$ (in rosso)
e infine il grafico di $y=\a x^2+\b$
(in verde):
\begin{center}
\begin{tikzpicture}
\begin{axis}[xlabel=$x$,ylabel =$y$,
ylabel style={rotate=-90},
cycle list={blue,red,green,orange,cyan},
axis x line=center,axis y line=center,
ymin=0,ymax=19,xmin=-1.5,xmax=1.5,
smooth]
 \plotit{\function}{first}
 \plotit{\a*\function}{second}
 \plotit{\a*\function+\b}{third}
\end{axis}
\end{tikzpicture}
\begin{itemize}
\legendit{first}{}{}{blue}
\legendit{second}{\a}{}{red}
\legendit{third}{\a}{\b}{green!50!lime}
\end{itemize}
\end{center}
Cliccando sui pulsanti con la $x$ posti
di fianco  a ciascuna funzione, è
possibile visualizzarne o no il grafico.
\end{solution}}
```

I comandi `\function`, `\button`, `\plotit` e `\legendit` vengono definiti nel file master (usato per la compilazione dell'esercitazione da somministrare agli studenti), in modo del tutto analogo a quanto visto nel primo esempio del paragrafo 3.3.2:

```
\newcommand{\function}{\x^2}
\newcommand{\button}[2]{\tikz[base]%
\node[fill=#2!30,ocg button=#1]{};}
\newcommand{\plotit}[2]{\addplot+%
[ocg={name=#2,ref=#2}]{#1};\label{#2}}
\newcommand{\legendit}[4]{\ifx&#3&%
\item[\ref{#1}] #2$x^2$ \button{#1}{#4}
\else
\item[\ref{#1}] #2$x^2+$#3
\button{#1}{#4}
\fi}
```

Il codice presentato produce un file interattivo, nel quale agli studenti è dapprima presentato il grafico della funzione $y = ax^2 + b$, con al posto di a e b due valori numerici scelti a caso tra quelli indicati nel comando `\FPsetpar`. Lo studente sceglie la risposta che ritiene corretta tra quelle proposte, clicca il pulsante "Fine Quiz" e può visualizzare solo allora la risposta corretta e la traccia di soluzione che il docente ha scritto (nell'ambiente solution). Nella soluzione, il grafico interattivo permette allo studente di visualizzare i passaggi che portano alla rappresentazione del grafico proposto.

5 Conclusioni

Sia le animazioni che i grafici interattivi sono strumenti indubbiamente efficaci dal punto di vista didattico, perché aiutano gli studenti a visualizzare gli "oggetti" di studio. Per le animazioni LaTeX fornisce strumenti efficaci e collaudati ed è compito dell'autore scegliere quello che ritiene più appropriato. Invece, per quanto riguarda i grafici interattivi molto dipende dal codice "esterno" che deve essere utilizzato, in particolare Javascript, che obbliga all'utilizzo di Adobe Acrobat Reader come lettore dei file `pdf`. In alcuni casi, purtroppo, ci sono problemi di compatibilità con i pacchetti utilizzabili per la creazione di presentazioni e di esercizi. Inoltre l'utilizzo del codice Javascript limita fortemente la possibilità di personalizzazione (caratteri, colori, ecc.) degli oggetti grafici che vengono creati. È nostra intenzione sperimentare ampiamente questi strumenti nel materiale didattico che mettiamo a disposizione degli studenti al fine di valutarne la reale efficacia e tentare di risolvere alcuni dei problemi che abbiamo incontrato, in particolare di compatibilità con il pacchetto minerva.

Riferimenti bibliografici

Baltieri, N. e Scandola, M. (2014). «Scrivere report statistici con R e Sweave». *ArsTEXnica*, **18**, pp. 32–38.

Bock, R., Linares, P. e Toro, J. (2014). «Il pacchetto interactiveplot». `CTAN:macros/latex/contrib/interactiveplot`.

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

De Marco, A. e Giacomelli, R. (2011). «Creare grafici con pgfplots». *ArsTEXnica*, **12**, pp. 9–35.

Drake, D. *et al.* (2009). «The SageTEX package». `CTAN:macros/latex/contrib/sagetex`.

Feuersänger, C. (2015). «Manual for package pgfplots». CTAN:macros/latex/contrib/pgfplots.

Grahn, A. (2012). «Il pacchetto movie15». CTAN:macros/latex/contrib/movie15.

— (2015a). «Il pacchetto animate». CTAN:macros/latex/contrib/animate.

— (2015b). «Il pacchetto media9». CTAN:macros/latex/contrib/media9.

Huss, W. e Maříк, R. (2009). «sws2tex». https://bitbucket.org/whuss/sws2tex/src.

Isambert, P. e Gaborit, P. (2012). «The ocgx package (version 0.5)». CTAN:macros/latex/contrib/ocgx.

Messineo, G. e Vassallo, S. (2012). «Test online di matematica: il progetto M.In.E.R.Va.» *ArsTEXnica*, **14**, pp. 104–112.

— (2013). «Il pacchetto minerva». *ArsTEXnica*, **16**, pp. 58–67.

Tantau, T. (2013). «The TikZ and PGF Packages». http://sourceforge.net/projects/pgf.

Van Zandt, T. (2010). «Documentation for multido.tex, version 1.42: a loop macro for generic TEX». CTAN:macros/latex/contrib/multido.

▷ Grazia Messineo
Università Cattolica Milano – IIS "Falcone-Righi" Corsico
grazia dot messineo at unicatt dot it

▷ Salvatore Vassallo
Università Cattolica Milano
salvatore dot vassallo at unicatt dot it