Creare grafici con **pgfplots**

Agostino De Marco, Roberto Giacomelli

Sommario

In questo articolo viene presentato PGFPLOTS, il pacchetto per il disegno di grafici basato su PGF. I manuali d'uso di questi due pacchetti di estensione del linguaggio LaTeX sono molto voluminosi e dettagliati e spesso gli utenti sono scoraggiati dall'affrontare lo studio di questi due documenti. In particolare, quelli desiderosi di creare grafici tecnico-scientifici di alta qualità tipografica hanno difficoltà a districarsi nei dettagli delle tante opzioni e personalizzazioni possibili. L'impostazione dell'articolo suggerisce un approccio pratico: si mostra come preparare un grafico a partire dall'impostazione degli assi, introducendo così i comandi e le opzioni fondamentali; successivamente si introducono le nozioni per il disegno di curve e superfici, spiegando come personalizzarne l'aspetto.

Abstract

This article presents PGFPLOTS, the package for the creation of plots based on PGF. The user manuals of both these extensions of the LaTeX language are very detailed and voluminous and users are often discouraged from addressing the study of these two documents. In particular, those wishing to create high quality graphs have a difficult time to disentangle the details of the many options and customizations. The article suggests a practical approach: It will be shown how a graph can be prepared by means of a correct initial setting of axes, thus introducing the basic commands and options. Then the way to plot curves and surfaces will be explained, suggesting how to possibly customize their appearence.

1 I grafici e la tipografia

La *rappresentazione grafica* dei dati numerici è un'attività essenziale nella comunicazione tecnico-scientifica. Nelle scienze e nell'ingegneria è importante facilitare la comprensione tanto dei fenomeni fisici quanto dei modelli matematici. I fenomeni fisici vengono descritti con l'ausilio delle risultanze di rilievi sperimentali, nei quali si provvede al collezionamento di dati in forma numerica. Nella fisica matematica, in cui si lavora essenzialmente con modelli matematici che rappresentano un'idealizzazione dei sistemi fisici reali, si ha la necessità di evidenziare le dipendenze di alcune grandezze fisiche in funzione di altre. In matematica spesso si ricorre all'uso di opportune visualizzazioni grafiche per far comprendere con più immediatezza alcuni aspetti teorici od apprezzare l'aspetto complessivo dell'intero insieme di dati. Si ha così che l'elaborazione e la rappresentazione dei dati — cioè il tracciamento di curve, superfici e altri tipi di grafici — è, di fatto, una parte integrante del lavoro tecnico-scientifico.

Come tutti gli elementi di un manoscritto, anche i *grafici* devono rispettare le regole tipografiche generali per offrire al lettore la fondamentale chiarezza dei contenuti e l'irrinunciabile consistenza del documento. Proprio questo è lo scopo del pacchetto PGFPLOTS, quello cioè di offrire all'utente LaTeX uno strumento per disegnare grafici di alta qualità che rispettino le scelte tipografiche alla base del lavoro che si sta scrivendo.

Basta sfogliare una rivista o un libro per renderci conto dei numerosi e svariati tipi di visualizzazioni numeriche esistenti nella comunicazione tecnico-scientifica moderna. Troveremo un gran numero di modi di rappresentazione e particolarità di formato. Oggi ci si aspetta da un programma informatico in grado di disegnare grafici che esso: (*i*) offra la possibilità di impostare le proprietà di colore, lo spessore e il tipo linea, il font degli elementi testuali, il posizionamento relativo dei vari oggetti grafici, (*ii*) permetta di tracciare sia curve che superfici, (*iii*) consenta di aggiungere legende e titoli, (*iv*) permetta di personalizzare a piacimento gli assi di riferimento e le griglie.

2 Il pacchetto **pgfplots**

Il pacchetto PGFPLOTS tenta di offrire tutto questo nell'ambiente testuale dei sorgenti dei sistemi TeX, ovvero per mezzo di un *linguaggio*. Esso sta riscontrando un successo crescente presso gli utenti principalmente perché fornisce la possibilità di realizzare molte possibili varianti di rappresentazione spesso raffinate e di precisione. Come si vedrà più avanti e come assicurano gli utilizzatori entusiasti del pacchetto, il linguaggio introdotto da PGFPLOTS è percepito dall'utente come un linguaggio naturale ed efficiente e ciò contribuirà certamente alla sua fortuna.

Per ottenere le caratteristiche elencate nella sezione precedente — un compito decisamente impegnativo — non sorprende che come libreria di base per l'implementazione di PGFPLOTS sia stato scelto il pacchetto PGF di Till Tantau (Tantau, 2010), un vero e proprio *motore* grafico ricchissimo sia di funzionalità che di concetti linguistici pronti all'uso (stili, opzioni chiave/valore, eccetera).

Una importante caratteristica di PGFPLOTS è la possibilità di effettuare calcoli numerici[1] attraverso le funzionalità fornite dalla libreria PGFmath sviluppata per le necessità del disegno.

L'Autore di PGFPLOTS è Christian Feuersänger che continua lo sviluppo del pacchetto curando in maniera particolare anche la documentazione del pacchetto (FEUERSÄNGER, 2010) e intervenendo sui forum specializzati[2]. Si rimanda ai manuali d'uso di PGFPLOTS e di PGF per qualsiasi approfondimento su concetti non trattati o trattati solo superficialmente in questo articolo.

3 Motori compatibili con pgfplots

Possiamo utilizzare il pacchetto PGFPLOTS scrivendo codice sia in linguaggio LaTeX sia in linguaggio ConTeXt. È anche possibile utilizzare PGFPLOTS scrivendo codice nel linguaggio TeX per l'interpretazione diretta del disegno dei grafici da parte del motore di composizione.

I formati LaTeX e ConTeXt prevedono una diversa sintassi di base quindi, a seconda della modalità di lavoro utilizzata, appariranno diverse le funzioni per costruire i grafici. La traduzione è tuttavia diretta e non si faticherebbe a passare dall'uno all'altro linguaggio perché i concetti sono immediatamente riconoscibili.

In questo articolo forniamo esempi in LaTeX, dunque per caricare il pacchetto nel documento useremo il comando `\usepackage` anziché la direttiva `\usemodule` prevista in ConTeXt. Inoltre, in LaTeX l'ambiente fondamentale è axis mentre in ConTeXt va usata la coppia di comandi `\startaxis` e `\stopaxis` per racchiudere il codice di un grafico.

4 Disegnare un sistema di assi di riferimento con l'ambiente axis

In LaTeX l'elemento sintattico di base del pacchetto PGFPLOTS è l'ambiente axis. Esso deve a sua volta essere inserito in un ambiente tikzpicture messo a disposizione dal pacchetto padre PGF. Lo schema sintattico è perciò il seguente:

```
% nel preambolo
\usepackage{pgfplots}% carica anche tikz
...
\begin{tikzpicture}
   \begin{axis}[⟨opzioni grafiche⟩]
   ...
   ⟨comandi di pgfplots o di tikz⟩
   ...
   \end{axis}
\end{tikzpicture}
```

In termini pratici, un disegno realizzato con PGFPLOTS — cioè un ambiente axis — è visibile all'utente LaTeX come un componente grafico da utilizzare all'interno di una illustrazione generata tramite il pacchetto tikz — cioè un ambiente tikzpicture[3].

Tabella 1: Tipi di grafico disponibili in PGFPLOTS

Tipo di grafico	Ambiente in PGFPLOTS
Piano cartesiano	axis
Piano logaritmico	loglogaxis
Ascissa logaritmica	semilogxaxis
Ordinata logaritmica	semilogyaxis
Piano polare	polaraxis
Diagrammi ternari	ternaryaxis
Diagrammi di Smith	smithchart

L'ambiente axis è quello fondamentale e serve a rappresentare entità grafiche nel sistema di riferimento cartesiano noto dalle scuole di base. Ai grafici logaritmici o polari corrispondono in PGFPLOTS altri ambienti specifici che sono elencati nella tabella 1. Nelle fasi di lavorazione di un grafico la personalizzazione di questi ambienti attraverso opportune opzioni è l'attività più importante.

4.1 Uso minimale dell'ambiente axis

Ecco un esempio di codice davvero minimale:

```
\begin{tikzpicture}
   \begin{axis}
   \end{axis}
\end{tikzpicture}
```

Esso produce il disegno mostrato nella figura 1a utilizzando delle impostazioni predefinite. Come si vede, in mancanza di direttive e di dati o funzioni specifiche da rappresentare, i valori minimi e massimi di entrambi gli assi coordinati (che definiscono la cosiddetta *tela* del disegno) sono pari, rispettivamente, a −0,1 e 1,1. Inoltre le coppie di tacche (*tick mark*) consecutive su ciascun asse coordinato individuano segmenti di misura 0,2.

4.2 Personalizzazione dell'ambiente axis

Queste proprietà possono essere personalizzate fornendo delle opportune opzioni all'ambiente axis. Eccone un esempio:

```
\begin{axis}[
   xmin = -1, xmax = 1,
   ymin =  0, ymax = 2,
   grid = major,
   xlabel = $x$, ylabel = $y$
]
\end{axis}
```

Questo codice produce il disegno mostrato nella figura 1b. In generale le opzioni vengono passate secondo lo schema ⟨*chiave*⟩=⟨*valore*⟩.

1. Per le informazioni sulle capacità di calcolo di TeX si veda l'articolo (GIACOMELLI, 2008).

2. Si veda ad esempio il forum dedicato a TeX sul circuito *Stackexchange*: `http://tex.stackexchange.com/`.

3. Il codice del pacchetto PGF è strutturato in più livelli. Il modulo più esterno, che implementa il linguaggio per l'utente si chiama tikz, ecco il motivo per cui in questo articolo si fa riferimento ad entrambi i nomi.

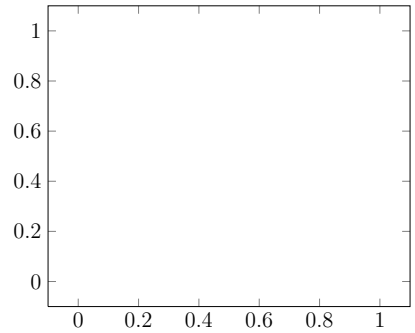
Nel codice precedente le chiavi xmin e xmax fissano il valore minimo e il valore massimo delle ascisse. Analoghe chiavi vengono adoperate per l'asse delle ordinate. L'utente deve tener presente che quando ha esigenze particolari sull'impostazione dei limiti di un asse (ad esempio delle ordinate) è tenuto a fornire esplicitamente anche i limiti dell'altro asse (delle ascisse); questo per dare la possibilità a PGFPLOTS di modificare automaticamente alcune impostazioni di visualizzazione collegate.

L'opzione grid permette di inserire una griglia, in questo caso a partire dalle tacche di marcatura principale degli assi, destinata a facilitare la lettura di un eventuale grafico.

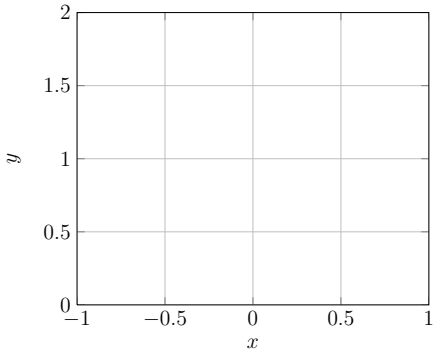
Le opzioni xlabel e ylabel consentono di inserire le etichette degli assi, in questo caso corrispondenti a x e y. Va osservato che l'impostazione predefinita riguardante l'etichetta dell'asse delle ordinate corrisponde alla prassi molto diffusa secondo la quale essa viene ruotata di 90° e centrata verticalmente rispetto all'estensione dell'asse.

Si prenda in esame ora la figura 1c. Essa presenta un espediente che si rivela utile ogni volta che si deve scalare un'immagine all'interno di un manoscritto e al tempo stesso si vogliono rendere ben visibili le annotazioni in essa presenti. In questi casi va sempre ricordato che anziché scalare indiscriminatamente una figura — che si tratti di un'illustrazione qualitativa o di un grafico — essa dovrebbe piuttosto essere progettata sempre in funzione di una coppia ben fissata di dimensioni finali. Tuttavia, dato che spesso si lavora per aggiustamenti ciclici del contenuto di un manoscritto e al tempo stesso delle immagini che esso contiene, non è raro dover ritoccare la scalatura di qualche figura e cambiare la dimensione del carattere delle sue annotazioni per ottimizzare la leggibilità. Per questa esigenza viene in aiuto il pacchetto relsize. La figura 1c si distingue dalla 1b perché presenta delle etichette degli assi di dimensione maggiore. Ciò è ottenuto con il codice seguente che fa uso del comando `\relsize`:

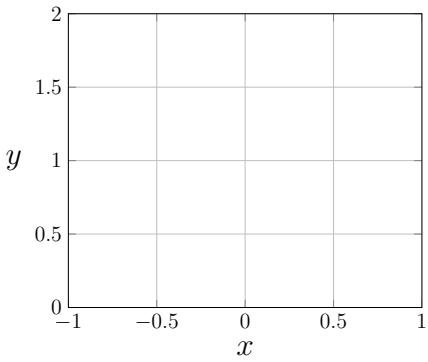
```
% nel preambolo
\usepackage{pgfplots,relsize}
...
% impostazione degli stili di pgfplots
\pgfplotsset{
    every axis x label/.append style = {
       font = \relsize{1}
    },
    every axis y label/.append style = {
       font = \relsize{1},
       rotate = -90,
       xshift = 0.5em
    }
}
\begin{tikzpicture}
\begin{axis}[xmin = -1, xmax = 1,
             ymin =  0, ymax = 2,
             grid, xlabel = $x$, ylabel = $y$
]
\end{axis}
\end{tikzpicture}
```



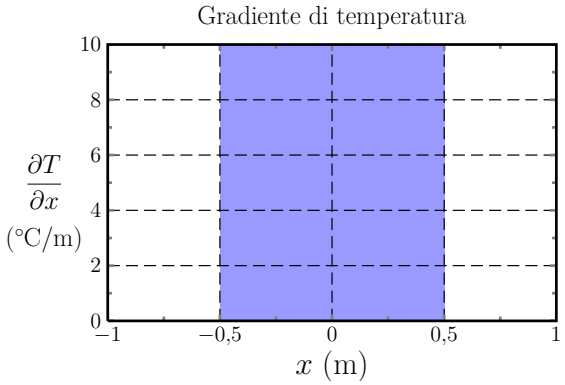
(a) Uso minimale dell'ambiente axis.



(b) Personalizzare gli estremi degli assi e le loro etichette.



(c) Regolare le dimensioni delle etichette degli assi con il pacchetto relsize e rotazione dell'etichetta dell'asse delle ordinate.



(d) Inserire un titolo; cambiare lo spessore delle linee di riferimento; usare il pacchetto siunitx; regolare la posizione dell'etichetta dell'asse delle ordinate.

FIGURA 1: Rappresentazioni ottenute con il solo l'ambiente axis, al quale vengono passate successivamente alcune opzioni di configurazione.

Questo è un primo esempio di utilizzo degli *stili* di PGFPLOTS. Il primo stile che viene modificato nel codice precedente è `every axis x label/` che, come si intuisce dal nome, si riferisce allo stile predefinito dell'etichetta dell'asse delle ascisse.

Gli stili sono delle proprietà che l'utente usa come delle variabili. Se ne può impostare un valore diverso da quello predefinito al fine di personalizzare una data caratteristica del disegno. Il concetto di stile viene introdotto dal pacchetto padre PGF. Uno stile viene percepito dall'utente come un percorso in un albero di proprietà (molto simile al percorso di un file in un albero di cartelle e sottocartelle). Uno stile di PGF viene cambiato attraverso il comando `\pgfkeys`. PGFPLOTS introduce degli stili aggiuntivi che sono modificabili attraverso la macro `\pgfplotsset`.

Nel codice precedente, prima di tutte le occorrenze dell'ambiente axis, viene alterata la dimensione predefinita delle etichette degli assi orizzontali (etichetta "x" nella figura 1c) con la direttiva seguente: `every axis x label/.append style`. L'azione `append style` viene applicata alla specifica proprietà; le assegnazioni che essa comporta vengono *aggiunte* a quelle predefinite. Tipicamente nelle direttive di modifica dello stile compare una lista di parole chiave corrispondenti ad altrettante proprietà personalizzabili. Nel caso specifico la dimensione del carattere viene cambiata assegnando alla proprietà font il valore `\relsize{1}`. L'effetto è quello di aumentare la dimensione del carattere di una *unità* rispetto a quella corrente (ad esempio, se la dimensione corrente è `\normalsize` il carattere diventerà di dimensione `\large`; `\relsize{-1}` farebbe passare la dimensione al valore `\small`).

In maniera analoga nel codice precedente si va ad aumentare la dimensione dell'etichetta y. Quest'ultima viene anche ruotata con l'assegnazione rotate=-90 e spostata verso destra con l'assegnazione xshift=0.5em.

Infine, si esamini la figura 1d che evolve dalla precedente e si ottiene con il codice seguente:

```
% nel preambolo
\usepackage{pgfplots,relsize,siunitx}
...
% impostazione di uno stile di pgf
\pgfkeys{
      /pgf/number format/.cd, % "change directory"
      use comma
}
% impostazione di stili di pgfplots
\pgfplotsset{
   every axis/.append style = {
      font=\relsize{-1},
         % riguarda le tick labels
      line width = 1.2pt,
         % oppure: thin, semithick, thick,
         %   very thick
      tick style = {line width = 1.2pt}
      },
   every axis x label/.append style = {
      font = \relsize{1}
   },
   every axis y label/.append style = {
      font = \relsize{1},
      rotate = -90,
      xshift = -0.7em,
      yshift = -1.4em
   },
   major grid style = {
      line width = 0.8pt,
      black,
      dash pattern = on 8pt off 4pt
   },
   every axis title/.append style = {
      font = \relsize{1}
   }
}
\begin{tikzpicture}
\begin{axis}[
   width = 12cm, height = 8cm,
   xmin = -1, xmax = 1,
   ymin =  0, ymax = 10,
   xtick = {-1,-0.5,...,1},
   ytick = {0,2,...,10},
   minor x tick num = 1,
   minor y tick num = 1,
   grid = major,
   xlabel = {$x$ (\si{\meter})},
   ylabel={
      \parbox{2cm}{
               \centering
               $\dfrac{\partial T}{\partial x}$
               \\[0.7em]
               \centering
               (\si{\celsius/\meter})
      }
   },
   title = Gradiente di temperatura,
   axis on top = true
]
% rettangolo
\fill[blue!40]
    (axis cs:-0.5,0) --
    (axis cs:0.5,0) --
    (axis cs:0.5,10) --
    (axis cs:-0.5,10) --
    cycle;
\end{axis}
\end{tikzpicture}
```

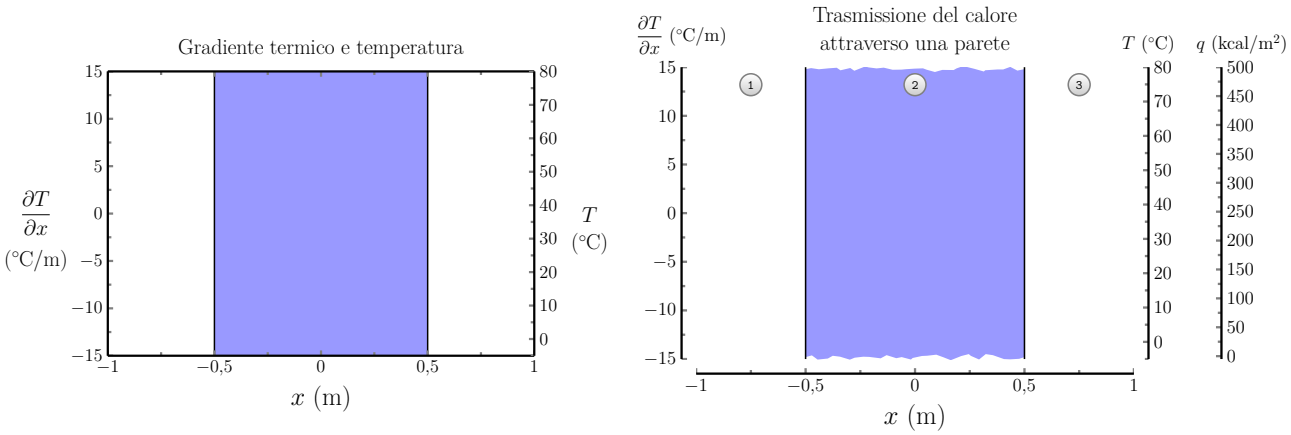
Le personalizzazioni introdotte nella figura 1d sono le seguenti:

(*i*) si è impostata la virgola come separatore decimale nei valori numerici — allo stile `/pgf/number format/` di PGF si applica l'azione denominata `.cd`; essa applica la direttiva predefinita `use comma` (si veda il manuale d'uso del pacchetto PGF per approfondimenti);

(*ii*) sono state impostate esplicitamente la larghezza e l'altezza complessive occupate dal grafico — questa è una tecnica di scalatura che attraverso le opzioni width e height *non* determina cambiamenti nelle dimensioni delle etichette testuali;

(*iii*) le etichette degli assi sono corredate di dimensioni — grazie al pacchetto siunitx, che fornisce il comando `\si`, si è potuto comporre in maniera appropriata l'etichetta dell'asse delle ascisse, che è ora: "x, (m)", cioè una dimensione lineare;

(*iv*) l'etichetta dell'asse delle ordinate è ruotata di 90° e, per non occupare molto spazio in larghez-



(a) Due assi delle ordinate contrapposti.

(b) Assi delle ordinate multipli.

FIGURA 2: Ottenere assi delle ordinate con differenti fattori di scala.

za, è composta come un capoverso su due righe — viene usata la macro `\parbox`;

(*v*) è stato assegnato uno spessore di 1,2 pt alle linee degli assi e ai *tick mark* — modificando lo stile `every axis/`;

(*vi*) sono stati impostati manualmente i *tick mark* su entrambi gli assi — si vedano le assegnazioni riguardanti le opzioni xtick e ytick;

(*vii*) per ogni coppia di *tick mark* principali consecutivi su entrambi gli assi è stata aggiunta una tacca secondaria — si vedano le assegnazioni riguardanti le opzioni minor x tick num e minor y tick num;

(*viii*) si è assegnato uno spessore di 0,8 pt, un colore nero e un tratteggio discontinuo alle linee della griglia — si veda l'impostazione della proprietà major grid style;

(*ix*) si è definito un un titolo del grafico, posto in alto e centrato orizzontalmente — ottenuto con l'opzione title;

(*x*) si è disegnata un'area rettangolare colorata avente un'attinenza con le grandezze fisiche rappresentate — si veda il comando `\fill` del pacchetto PGF; l'assegnazione axis on top=true fa in modo che gli assi e la griglia vengano disegnati per ultimi sovrapponendosi ai restanti elementi grafici.

4.3 Uso avanzato dell'ambiente axis

Illustriamo qui un esempio di uso avanzato dell'ambiente axis facendo vedere come si realizzano le figure 2a e 2b. Entrambe si basano sulla possibilità di porre in uno stesso ambiente tikzpicture più istanze dell'ambiente axis.

Vediamo inizialmente come si realizza la figura 2a. Non si riporterà tutto il codice necessario, che è simile a quello riportato in precedenza e con il quale si è generata la figura 1d. Il codice seguente mostra come si può ottenere un secondo asse delle ordinate, posto a destra della figura e con diverse unità di misura:

```
\begin{tikzpicture}
% nel preambolo
\pgfplotsset{compat=1.3}
...
% impostazioni condivise
\pgfplotsset{
   scale only axis,
   width=12cm, height=8cm,
}
% . . . . . . . . . . . . . . . . . . . . . . . . . . . . . axis #1
\begin{axis}[
   xmin=-1, xmax=1,
   ymin=-15, ymax=15,
   axis y line*=left,
   xtick={-1,-0.5,...,1},
   ytick={-15,-10,...,15},
   minor x tick num = 1,
   minor y tick num = 1,
   xlabel={$x$ (\si{\meter})},
   ylabel={
      \parbox{2cm}{%
               \centering
               $\dfrac{\partial T}{\partial x}$
               \\[0.7em]
               \centering
               (\si{\celsius/\meter})
      }
   },
   title=Gradiente termico e temperatura,
   axis on top=true
]
% rettangolo colorato (si veda il manuale di pgf)
\fill[blue!40]
    (axis cs:-0.5,-15) --
    (axis cs:-0.5,15) --
    (axis cs:0.5,15) --
    (axis cs:0.5,-15) --
    cycle;
\draw[very thick]
   (axis cs:-0.5,-15) -- (axis cs:-0.5,15);
\draw[very thick]
   (axis cs:0.5,-15) -- (axis cs:0.5,15);
%
% Disegnare qui la curva del gradiente termico ∂T/∂x
% ...
\end{axis}
% impostazioni successive
\pgfplotsset{
   every axis y label/.append style={
      % aggiusta il posizionamento dell'etichetta destra
```

```
      xshift= -1.8em
   }
}
% . . . . . . . . . . . . . . . . . . . . . . . . . . . . . axis #2
\begin{axis}[
   xmin=-1, xmax=1,
   ymin=-5, ymax=80,
   hide x axis,
   axis y line*=right,
   ytick={-10,0,...,80},
   minor y tick num = 1,
   ylabel={
      \parbox{2cm}{%
               \centering
               $T$\\
               \centering
               (\si{\celsius})
      }
   },
]
%
% Disegnare qui la curva della temperatura T
% ...
\end{axis}
\end{tikzpicture}
```

In pratica due ambienti axis nello stesso ambiente tikzpicture risultano sovrapposti. Come si vede dal codice precedente nel primo ambiente axis è già predisposto il punto in cui eventualmente inserire il comando necessario a tracciare la curva del gradiente di temperatura $\partial T/\partial x$, cioè un elemento grafico disegnato nella scala indicata dalle marcature dell'asse delle ordinate sinistro.

Al secondo ambiente axis, che viene a porsi al di sopra del primo, viene fornita la direttiva hide x axis che nasconde l'asse delle ascisse. Inoltre la direttiva axis y line*=right permette di marcare soltanto l'asse delle ordinate posto dalla parte destra dell'immagine. Le opzioni ytick e ylabel del secondo ambiente axis determinano la nuova disposizione dei *tick mark* e la nuova etichetta. Nel codice è già predisposto anche il punto in cui eventualmente inserire il comando necessario a tracciare la curva della temperatura T.

La soluzione appena esaminata è utile quando si vogliono riportare più curve in uno stesso grafico. Ad esempio, la prima curva potrebbe dover essere disegnata nell'unità specificata dalla marcatura dell'asse delle ordinate sinistro; le restanti curve potrebbero poi essere disegnate seguendo l'unità definita dalla marcatura dell'asse destro. In tal caso la prima curva richiederà dei comandi di tracciatura all'interno del *primo* ambiente axis (si veda il comando `\addplot` più avanti). Le restanti curve richiederanno comandi di tracciatura posti all'interno del *secondo* ambiente axis.

A questo punto, vediamo come si generalizza la soluzione precedente. Spesso si ha l'esigenza di tracciare più curve aventi in comune la scala e i limiti dell'asse delle ascisse ma con valori delle ordinate molto diversi tra loro. In questi casi è preferibile riportare più assi delle ordinate opportunamente marcati e posizionati da una parte e dall'altra dell'immagine, in modo da poter mostrare con efficacia tutte le curve in una stessa area rettangolare. Un esempio è dato dalla figura 2b. Vediamo il codice che potrebbe consentire una simile rappresentazione:

```
% nel preambolo
\usepackage{pgfplots}
\usepgflibrary{decorations.pathmorphing}
\usepgflibrary{shapes.geometric,shapes.misc}
\pgfplotsset{compat=1.3}
...
\begin{tikzpicture}
% impostazioni di pgf (separatore decimale e delle migliaia)
\pgfkeys{
   /pgf/number format/.cd,
      set decimal separator={,{\!}},
      set thousands separator={}
}
% impostazioni generali di pgfplots
\pgfplotsset{
   width=12cm, height=8cm,
   scale only axis,
   no markers
}
% . . . . . . . . . . . . . . . . . . . . . . . . . . . . . axis #1a
%                                          l'asse y a sinistra
\begin{axis}[
   clip=false,% non taglia gli oggetti fuori dalla tela
   width=2cm, xshift=-0.4cm,% stringe e sposta
   hide x axis,
   axis y line*=left,
   xmin=-1, xmax=1,% necessario
   ymin=-15, ymax=15,
   ytick={-15,-10,...,15},
   minor y tick num=1
]
% Etichetta dell'asse posta 'a mano', in alto
\node [above, yshift=6pt]
   at (rel axis cs:0,1) {$\dfrac{\partial T}{
      \partial x}$ (\si{\celsius/\meter})};
% NON deve contenere altri comandi
\end{axis}
% . . . . . . . . . . . . . . . . . . . . . . . . . . . . . axis #1b
%       elementi grafici nella scala dell'asse y a sinistra
\begin{axis}[
   hide x axis,
   hide y axis,
   xmin=-1, xmax=1,
   ymin=-15, ymax=15
]
%
% Disegnare qui le curve in questa scala
% ...
\end{axis}
% . . . . . . . . . . . . . . . . . . . . . . . . . . . . . axis #2
%                                          l'asse x in basso
\begin{axis}[
   height=2cm, yshift=-0.4cm,% stringe e sposta
   axis x line*=bottom,
   hide y axis,
   xmin=-1, xmax=1,
   ymin=-15, ymax=15,% necessario
   xtick={-1,-0.5,...,1},
   minor x tick num=1,
   xlabel={$x$ (\si{\meter})}
]
% NON deve contenere comandi
\end{axis}
% . . . . . . . . . . . . . . . . . . . . . . . . . . . . . axis #3a
%                                    il primo asse y a destra
```

```
\begin{axis}[
   clip=false,
   xshift=0.4cm,% sposta
   hide x axis,
   axis y line*=right,
   xmin=-1, xmax=1,% necessario
   ymin=-5, ymax=80,
   ytick={-10,0,...,80},
   minor y tick num=1
]
% Etichetta dell'asse posta 'a mano', in alto
\node [above, yshift=6pt]
   at (rel axis cs:1,1) {$T$ (\si{\celsius})};
% NON deve contenere altri comandi
\end{axis}
% . . . . . . . . . . . . . . . . . . . . . . . . . . . . . axis #3b
%  elementi grafici nella scala del primo asse y a destra
\begin{axis}[
   hide x axis,
   hide y axis,
   xmin=-1, xmax=1,
   ymin=-15, ymax=15,
   title=\parbox{8cm}{\centering Trasmissione del
     calore attraverso una parete},
]
% la parete solida (si veda il manuale di pgf)
\fill[blue!40]
   decorate [decoration={random steps,segment
     length=2mm}]
     { (axis cs:-0.5,-14.8)
        -- (axis cs:0.5,-14.8)
     }
     -- (axis cs:0.5,14.8)
   decorate [decoration={random steps,segment
     length=2mm}]
     {
        -- (axis cs:-0.5,14.8)
     }
     -- cycle;
\draw[very thick] (axis cs:-0.5,-15)
   -- (axis cs:-0.5,15);
\draw[very thick] (axis cs:0.5,-15)
   -- (axis cs:0.5,15);
% zone '1', '2' e '3'
\node [rounded rectangle, minimum size=6mm, very
     thick, draw=black!50, top color=white, bottom
     color=black!20, font=\ttfamily]
   at (rel axis cs:0.125,0.94) {1};
\node [rounded rectangle, minimum size=6mm, very
     thick, draw=black!50, top color=white, bottom
     color=black!20, font=\ttfamily]
   at (rel axis cs:0.50,0.94) {2};
\node [rounded rectangle, minimum size=6mm, very
     thick, draw=black!50, top color=white, bottom
     color=black!20, font=\ttfamily]
   at (rel axis cs:0.875,0.94) {3};
%
% Disegnare qui le curve in questa scala
% ...
\end{axis}
% . . . . . . . . . . . . . . . . . . . . . . . . . . . . . axis #4a
%                              il secondo asse y a destra
\pgfplotsset{
   every axis y label/.append style={
      xshift=-1.3em
   }
}
\begin{axis}[
   clip=false,
   xshift=2.4cm,
   hide x axis,
   axis y line*=right,
   xmin=-1, xmax=1,% necessario
   ymin=-5, ymax=500,
   ytick={0,50,...,500},
   minor y tick num=1
]
% Etichetta dell'asse posta 'a mano', in alto
\node [above, yshift=6pt] at (rel axis
     cs:1,1) {$q$ (\si{\kcal/\meter^2})};
% NON deve contenere altri comandi
\end{axis}
% . . . . . . . . . . . . . . . . . . . . . . . . . . . . . axis #4b
% elementi grafici nella scala del secondo asse y a destra
\begin{axis}[
   xmin=-1, xmax=1,% necessario
   ymin=-5, ymax=500,
   hide x axis,
   hide y axis,
]
%
% Disegnare qui le curve in questa scala
% ...
\end{axis}
\end{tikzpicture}
```

I commenti e lo stile del codice su riportato evidenziano a sufficienza la struttura necessaria a realizzare un grafico a curve multiple riferito ad ordinate con scale multiple. Per disegnare un asse delle ordinate isolato l'idea di base è quella di creare un ambiente axis a sé (*i*) impostando i limiti in maniera analoga (si veda, ad esempio l'ambiente "1a" nel codice precedente) a quelli dell'ambiente axis utilizzato per disegnare la relativa curva (si veda l'ambiente "1b"), (*ii*) impostando eventualmente una larghezza ridotta con l'opzione width nel caso di asse posto dalla parte sinistra (ambiente "1a"), (*iii*) nascondendo l'asse delle ascisse con l'opzione hide x axis (ambiente "1b"), (*iv*) spostando lateralmente l'asse, a destra (ambienti "3a" e "4a") o a sinistra (ambiente "1a"), con l'opzione xshift, (*v*) nascondendo l'asse delle ordinate con l'opzione hide y axis nell'ambiente axis in cui si disegna la curva (ambienti "1b", "3b" e "4b").

5 Tracciare curve con il comando \addplot

All'interno dell'ambiente che corrisponde al tipo di rappresentazione — nel caso più frequente una rappresentazione in assi cartesiani ottenuta con axis — è disponibile il comando \addplot che consente di aggiungere sulla tela una curva. In una singola rappresentazione sarà possile aggiungere tante curve con altrettanti comandi \addplot.

Il disegno di una curva in PGFPLOTS è prodotto con la linea spezzata che unisce la sequenza di punti definita in una delle modalità elencate fra poco. Punti sufficientemente vicini daranno una spezzata che sarà percepita visivamente come una curva continua e morbida raggiungendo così un buon grado di qualità e precisione del tracciamento.

Se tra le opzioni compare la chiave smooth il tracciamento si baserà invece su linee a curvatura

continua passanti per i punti stabiliti e definite matematicamente come polinomi, un abilità che ci ricorda le curve di Bézier di Metapost e del pacchetto Pstricks. Il disegno morbido della curva consentirebbe di diminuire la *densità* dei punti del grafico guadagnando in prestazioni ed ottenendo l'andamento desiderato ma introducendo un'ipotesi su come la curva vari tra due punti consecutivi non vera. Dovremo ricercare l'equilibrio tra precisione matematica e risorse di elaborazione.

L'utente potrà tracciare la curva in quattro modi possibili:

(*1*) attraverso l'uso di un'espressione matematica, che verrà valutata per un numero predefinito di punti all'interno di un dominio di definizione (numero comunque personalizzabile a piacimento attraverso un'opzione opportuna). A questa modalità corrisponde un codice avente la seguente struttura:

```
\begin{axis}% rappresentazione cartesiana
...
% grafico di una funzione
\addplot [⟨opzioni di disegno⟩] {
   ⟨espressione matematica⟩
};
...
\end{axis}
```

(*2*) attraverso una sequenza di coppie di coordinate, racchiuse tra parentesi tonde e separate da una virgola. A questa modalità corrisponde un codice avente la seguente struttura:

```
\begin{axis}% rappresentazione cartesiana
...
% spezzata che unisce una lista di punti
\addplot [⟨opzioni di disegno⟩] coordinates {
        ( x1 , y1 )
        ( x2 , y2 )
        ...
        ( xN , yN )
};
...
\end{axis}
```

(*3*) caricando direttamente da un file esterno di tipo testuale i valori numerici delle coordinate. Il formato atteso dei dati corrisponde a una sequenza di righe, in ciascuna delle quali sono riportati due o tre valori numerici separati da un carattere di tabulazione o da caratteri spazio. Le coppie o terne di valori corrispondono alle coordinate di punti del piano o dello spazio. A questa modalità corrisponde un codice avente la seguente struttura:

```
\begin{axis}% rappresentazione cartesiana
...
% spezzata che unisce una lista di punti
\addplot [⟨opzioni di disegno⟩] file
   {⟨nome del file dati⟩};
...
\end{axis}
```

(*4*) attraverso una sequenza di coppie di coordinate precedentemente memorizzate in una struttura dati attraverso il pacchetto PgfplotsTable. A questa modalità corrisponde un codice avente la seguente struttura:

```
% nel preambolo
\usepackage{pgfplots,pgfplotstable}
...
% Una struttura table definita esplicitamente (inline)
\pgfplotstableread{
        x1    y1
        x2    y2
        ...
        xN    yN
   }\mytable       % definisce questa nuova macro
...
\begin{axis}% rappresentazione cartesiana
...
% spezzata che unisce una lista di punti
\addplot [⟨opzioni di disegno⟩] table \mytable;
% la macro \mytable è riusabile
...
\end{axis}
```

Come si osserva dagli esempi precedenti, la sintassi di `\addplot` prevede il punto e virgola finale (come per gli altri comandi di PGF, trovandosi all'interno di un ambiente tikzpicture). Anche qui, analogamente ai parametri di formato e agli elementi generali del grafico configurabili attraverso le opzioni dell'ambiente axis, le proprietà delle curve si personalizzano con delle opportune opzioni del comando `\addplot`. In altri termini, le opzioni di `\addplot` sono le *opzioni di disegno* delle curve, assegnate anch'esse secondo lo schema ⟨*chiave*⟩=⟨*valore*⟩.

Come è affermato nel manuale d'uso del pacchetto, il comando `\addplot` è l'interfaccia principale tra l'utente e le funzionalità di PGFPLOTS per quanto riguarda le operazioni di tracciamento delle curve. D'altra parte, come si è visto nella sezione precedente, molte configurazioni importanti vengono effettuate tramite l'ambiente axis che racchiude le occorrenze di `\addplot`. Un livello ancora più esterno e generale di configurazione è offerto dai comandi `\pgfplotsset` e `\pgfkeys`.

5.1 Funzioni matematiche

Andiamo a vedere come deve essere usato il comando `\addplot` quando si vogliono definire curve attraverso espressioni matematiche.

Nello svolgere lo *studio di funzione* studenti ed insegnanti, ma anche matematici, fisici ed ingegneri, vanno a tracciare una curva definita da una data espressione matematica, la funzione. L'analisi della funzione viene eseguita con foglio e matita, determinando eventuali zeri, punti di discontinuità, asintoti, flessi, massimi e minimi. Il disegno della curva effettuato attraverso l'uso del computer serve spesso a confermare la correttezza dell'analisi. Andiamo dunque a disegnare con PGFPLOTS nel piano cartesiano xy la curva di equazione $y = f(x)$ di una funzione reale f di una variabile reale.

Disegnamo anche gli assi coordinati x ed y con tanto di freccia per indicarne il verso positivo, una

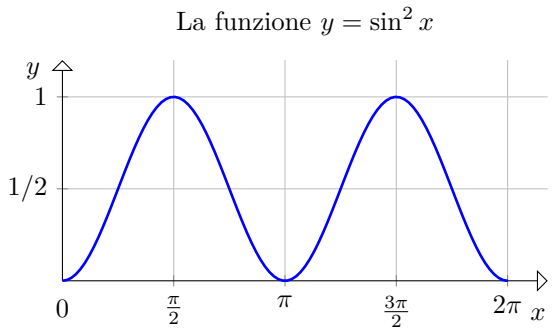


FIGURA 3: Esempio di studio di funzione

griglia di riferimento ed infine le etichette degli assi in stile matematico.

La funzione che si vuole rappresentare per questa prova è $f(x) = \sin^2 x$, periodica, di periodo π. Da un'analisi precedente si sa che il grafico della funzione, limitata all'intervallo $[(n-1)\pi, n\pi]$ (con $n = 1, , \ldots, \infty$), è simmetrico rispetto alla retta verticale di ascissa $(n - 1/2)\pi$. Tale circostanza è osservabile nella figura 3 per $n = 1$ ed $n = 2$. Ulteriore asse di emisimmetria delle creste dell'onda è la retta $y = 1/2$, che possiamo esprimere con la condizione $f(x) - 1/2 = -f(x - \pi/2) + 1/2$. In altre parole le creste d'onda al di sopra della retta orizzontale di ordinata $1/2$ hanno la stessa forma di quelle che si trovano sotto tale riferimento.

Il trattamento dell'espressione matematica che forniremo ad `\addplot` può essere impostato anche a monte, tra le opzioni dell'ambiente axis, valendo così per tutte le occorrenze del comando `\addplot` al suo interno. Se si osserva la curva della figura 3 si vede che essa è limitata ai valori della x appartenenti all'intervallo $[0, 2\pi]$. Ciò si ottiene definendo esplicitamente il dominio attraverso l'opzione `domain`. Dunque la parte essenziale del codice di disegno potrà assomigliare a qualcosa del genere:

```
\begin{axis}[
   domain=0:2*pi,     % 0 ≤ x ≤ 2π
   samples=100,       % dividi [0,2π] in 100 parti uguali
   xmin=0, xmax=6.85,
   ymin=0, ymax=1.20,
   ...
   ]
   \addplot[% opzioni di disegno
      color=blue, line width=1pt
      ] {
         sin(deg(x))^2 % espressione matematica
      };
\end{axis}
```

L'espressione matematica `sin(deg(x))^2` è facilmente comprensibile da chiunque abbia un minimo di esperienza di programmazione. Si assume che l'identificatore `x` sia una variabile simbolica appartenente all'intervallo definito con domain. L'espressione simbolica è valutata attraverso la libreria matematica fornita dal pacchetto PGF. L'opzione sample indica il numero di punti in cui verrà calcolato il valore numerico della funzione dividendo il dominio in parti uguali. Questa configurazione rappresenta perciò una sorta di precisione di tracciatura, e va attentamente regolata in modo da ottenere un andamento adeguato del grafico della funzione. Si osservi che le funzioni trigonometriche di PGFsi aspettano argomenti in gradi e *non* in radianti. Ciò rende necessario l'uso della funzione `deg()` che accetta un argomento in radianti e restituisce un risultato in gradi.

L'uso che si è fatto di domain permette di ricirdare che le opzioni di axis, quando queste comportano la specifica di estremi o di intervalli numerici, accettano anche delle espressioni simboliche (si veda `2*pi`, dove `pi` è un identificatore predefinito pari a π). Si osservi inoltre che le opzioni domain e samples possono essere usate direttamente come opzioni delle singole occorrenze del comando `\addplot`; in tal caso il loro valore sovrascrive, per quella particolare curva, quello impostato dall'ambiente axis.

Nel frammento di codice abbiamo inserito i valori che definiscono l'area del grafico (la *tela*) attraverso le opzioni xmin, xmax, ymin e ymax. Gli assi e gli altri elementi verranno disegnati con riferimento alla tela, mentre la funzione verrà disegnata con riferimento al dominio. Come si può osservare, la larghezza e l'altezza della tela in quest'esenpio sono leggermente maggiori, rispettivamente, delle estensioni del dominio e del codominio della funzione. Ciò serve a far posto alle annotazioni previste sugli assi coordinati.

Richiamiamo qui il significato delle opzioni axis x line e axis y line già utilizzate negli esempi della sezione precedente. Attraverso di esse gli assi vengono disegnati come segmenti orientati con una freccia che indica il verso positivo (e non viene disegnato il rettangolo di contorno della tela). Il disegno della freccia si può scegliere tra quelli disponibili nel pacchetto e tra quelli ulteriori caricati della libreria arrows di PGF tra i quali il tipo open triangle 90, scelto per la figura 3. È anche possibile costruire un tipo di freccia personalizzato con il comando `\declarearrows` di PGF. Si rimanda al manuale di PGF per maggiori approfondimenti.

Per gestire la marcatura degli assi si deve far uso della coppia di opzioni xtick e xticklabels, per specificare rispettivamente le coordinate di marcatura dell'asse delle x ed il testo dell'etichetta da apporvi. Un discorso simile vale per la marcatura dell'asse delle y.

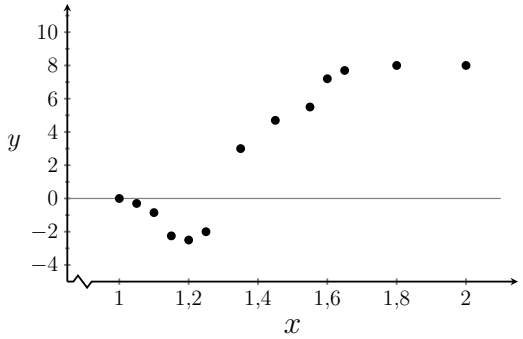
Il codice completo che genera il grafico della figura 3 è il seguente:

```
% nel preambolo
\usepackage{pgfplots}
\usepgflibrary{arrows}
...
\begin{tikzpicture}
\begin{axis}[
      domain=0:2*pi,      % dominio della funzione
      samples=100,        % campionamento
      xmin=0,             % definizione tela
```

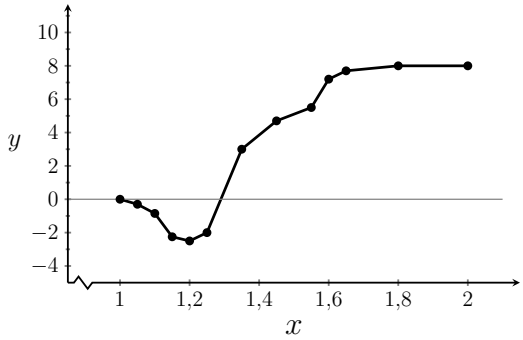
```
      xmax=6.85,
      ymin=0,
      ymax=1.20,
      axis x line=bottom, % posizione assi
      axis y line=left,
      % tipo di freccia
      every outer x axis line/.append style=
         {-open triangle 90},
      every outer y axis line/.append style=
         {-open triangle 90},
      % etichette
      title=La funzione $\sin^2 x$,
      xtick={1.5708,3.1416,4.7124,6.2832},
      xticklabels={$\frac{\pi}{2}$,
                   $\pi$,$\frac{3\pi}{2}$,$2\pi$},
      ytick={0.5,1},
      yticklabels={$1/2$,$1$},
      grid=major,
      width=6cm, height=3.5cm,% dimensionamento tela
      clip=false
]
   \addplot[color=blue, line width=1pt]
      sin(deg(x))^2; % espressione matematica

   % etichette posizionate 'a mano' di rifinitura
   \node at (axis description cs:0,-0.125){$0$};
   \node at (axis description cs:0.98,-0.15){$x$};
   \node at (axis description cs:-0.06,0.95){$y$};
\end{axis}
\end{tikzpicture}
```

Come si nota dal codice l'etichettatura degli assi è stata effettuata con apposite istruzioni di disegno `\node` anziché con le usuali opzioni xlabel e ylabel (per ottenere lo stesso effetto con queste ultime sarebbe stato necessario fornire delle direttive xshift e yshift agli stili delle etichette degli assi). Ciò dimostra, come già visto anche nella sezione precedente, che è sempre possibile utilizzare i comandi di disegno di PGF anche all'interno dell'ambiente axis; cosa utile per risolvere esigenze grafiche particolari. Occorre ricordare che per utilizzare questa possibilità va usata l'impostazione clip=false altrimenti tutto quello che si tenta di disegnare in un ambiente axis che cada graficamente all'esterno della tela verrà tagliato.

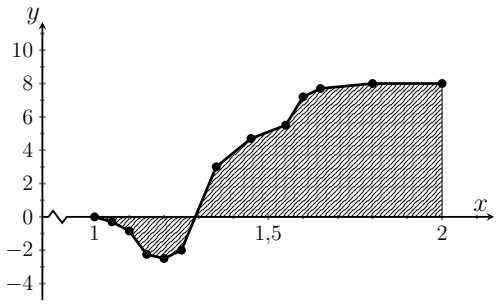
Si osservi che PGFPLOTS aiuta a trovare le coordinate dei punti nel grafico (dentro la tela o al di fuori di essa) mettendo a disposizione un certo numero di sistemi di riferimento appositamente definiti. I più usati sono individuati dai nomi: `axis cs` e `axis description cs`. È buona norma, quando si vogliono inserire segni e annotazioni aggiuntive in un grafico, effettuare sempre una scelta ragionata del riferimento in cui esprimerne la posizione. Per riferire una coppia di coordinate a un determinato sistema, se ne indica esplicitamente il nome seguito dai due punti e dalla coppia di numeri [ad esempio `(axis description cs: 0,-0.125)` come nel codice precedente]. Quando vi è certezza che le annotazioni sono interne alla tela il manuale di PGFPLOTS consiglia di utilizzare sempre il sistema `axis cs`. Si rimanda al manuale di PGFPLOTS per approfondimenti su questo argomento.



(a) uso di coordinate all'interno del comando `\addplot`, `\addplot [ only marks ] coordinates { ... };`



(b) uso di coordinate all'interno del comando `\addplot`, `\addplot [ ... ] coordinates { ... };`



(c) riuso di coordinate attraverso PGFPLOTSTABLE, `\addplot [ ... ] table \mytable;`

FIGURA 4: Esempio di spezzata che unisce un insieme di punti del piano

5.2 Lavorare con un insieme finito di dati

Vediamo ora come usare `\addplot` quando non si ha l'espressione analitica di una funzione ma un numero discreto di punti del piano.

Nelle scienze e nell'ingegneria si dispone spesso di una sequenza di N valori di una data grandezza fisica y in corrispondenza di altrettanti valori di un'altra grandezza fisica x che gioca il ruolo di variabile indipendente. Spesso questo tipo di corrispondenza è chiamato "funzione nota per punti" o "funzione tabellare". Una funzione può essere nota solo per punti perché deriva da misurazioni sperimentali oppure è soluzione numerica di un problema matematico.

Detti $P_i = (x_i, y_i)$, con $i = 1, \ldots, N$, gli N punti del piano, se si vuole rappresentare la funzione

tabellare in maniera semplice è sufficiente riportare in grafico una sequenza di segni grafici che danno l'idea di un *elemento concentrato* (pallini, quadratini, triangoli, stelle, eccetera). Un esempio è dato dalla figura 4a, dove $N = 13$. Spesso per dare un'idea dell'*andamento* dei dati si finisce per collegare le coppie P_iP_{i+1} (per $i = 1, \ldots, N-1$) con dei segmenti, come nella figura 4b.

Con PGFPLOTS ciò è immediato: basta elencare le coppie di coordinate all'interno del comando `\addplot`. Nel caso della prima figura si scrive:

```
\begin{tikzpicture}
\begin{axis}[
   xmin=0.85, xmax=2.15,
   ymin=-5, ymax=11.7,
   xtick={1,1.2,...,2},
   ytick={-4,-2,...,10},
   minor x tick num= 1,
   minor y tick num= 1,
   axis x line=bottom,
   axis y line=left,
   axis x discontinuity=crunch,
   xlabel={$x$}, ylabel={$y$}, title={},
   axis on top=true, clip=false
   ]
\addplot [mark=*,only marks] coordinates {
   (1.00,  0.00)
   (1.05, -0.30)
   (1.10, -0.85)
   (1.15, -2.25)
   (1.20, -2.50)
   (1.25, -2.00)
   (1.35,  3.00)
   (1.45,  4.70)
   (1.55,  5.50)
   (1.60,  7.20)
   (1.65,  7.70)
   (1.80,  8.00)
   (2.00,  8.00)
};
% retta orizzontale di ordinata 0
\draw[gray] (axis cs: 0.85,0) -- (axis cs: 2.1,0);
\end{axis}
\end{tikzpicture}
```

Si fornisce l'opzione only marks per ottenere il disegno dei soli punti e l'opzione mark=* per marcare i punti con un pallino pieno di dimensioni predefinite. La figura 4b si ottiene con un codice del tutto identico a quello precedente, nel quale però *non* viene specificata only marks tra le opzioni di `\addplot`.

L'esempio precedente è servito a mostrare un'interessante opzione dell'ambiente axis chiamata: axis x discontinuity. Nel caso in cui si ha necessità di rappresentare i dati in un'opportuna scala ma si vuole lo stesso far vedere l'origine, questa opzione permette di apporre un segno grafico di interruzione dell'asse delle ascisse (nell'esempio si è usato il tipo predefinito crunch). Una simile configurazione vale anche per l'asse delle ordinate.

La figura 4c è stata ottenuta dagli stessi dati dei precedenti due esempi ma la tecnica usata è leggermente più avanzata. Come di osserva, è stata evidenziata l'area compresa fra la spezzata che

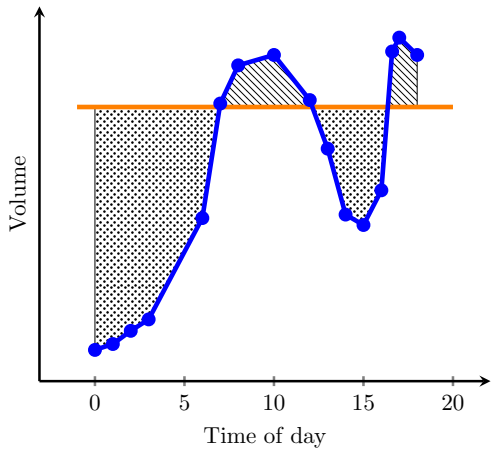


FIGURA 5: Esempio avanzato in cui si tratteggiano le varie regioni comprese fra una spezzata ed una retta orizzontale diversa dall'asse delle ascisse.

unisce i punti e l'asse delle ascisse (area *sottesa*). Quest'operazione richiede di dare due volte il comando `\addplot`, applicandolo allo stesso insieme di punti ma con opzioni di disegno diverse. Il codice che realizza il grafico in questione è il seguente:

```
% nel preambolo
\usepackage{pgfplots,pgfplotstable}
\usetikzlibrary{patterns} % per il tratteggio
...
% memorizza i dati una volta per tutte
\pgfplotstableread{
   1.00  0.00    % gli stessi dati del codice precedente
   1.05 -0.30
   1.10 -0.85
   ...
   2.00  8.00
   }\mytable
\begin{tikzpicture}
\begin{axis}[
   % ...
   axis x line=middle, % asse centrato verticalmente
   % ...
   ]
% disegna la spezzata e i pallini
\addplot [mark=*] table {\mytable};
% disegna l'area sottesa
\addplot [
   line width=0pt, % linea spezzata assente
   no marks,       % simboli assenti
   fill=black!70,  % area sottesa
   pattern=north east lines % riempimento
   ]                        % con linee oblique
   table \mytable \closedcycle; % chiude il tracciato
\end{axis}
\end{tikzpicture}
```

L'uso del comando `\pgfplotstableread` messo a disposizione dal pacchetto PGFPLOTSTABLE permette di memorizzare una volta per tutte le coppie di coordinate in un'opportuna struttura dati, che nell'esempio è chiamata `\mytable`. Essa viene usata due volte all'interno dell'ambiente axis fornendo al comando `\addplot` la direttiva `table`. La prima volta viene disegnata la spezzata che unisce i pallini. Successivamente viene disegnato il riempimento dell'area sottesa dalla curva tramite l'uso

dell'opzione fill. I commenti del codice precedente illustrano le altre peculiarità del disegno.

Un esempio di uso avanzato delle potenzialità combinate di PGF, PGFPLOTS e PGFPLOTSTABLE è riportato nella figura 5 [4].

5.3 Processare i dati all'esterno

Nel campo dell'elaborazione numerica sono oggi disponibili numerosi software, sia commerciali che gratuiti con i quali è possibile produrre dati e memorizzarli in molti formati di uso comune. Visualizzare questi risultati con PGFPLOTS è possibile sfruttando la direttiva `file` del comando `\addplot`.

Spesso questo procedimento richiede il trattamento di insiemi di punti molto numerosi. Per questi compiti più impegnativi — quando i valori sono generati con appropriati strumenti numerici di analisi e parallelamente si sta lavorando al manoscritto che deve riportare il grafico — è preferibile creare una cartella in cui spostare i file di dati e dedicare un sorgente LaTeX apposito per la creazione del grafico. Quest'ultimo sarà inserito nel documento finale sotto forma di immagine, ad esempio in formato PDF, attraverso il comando `\includegraphics`. In ogni caso è utile definire uno *stile* di disegno e di annotazione unico per tutti i grafici, congruente con lo stile tipografico del manoscritto (come per esempio le dimensioni del font delle etichette sugli assi o altri parametri costanti). Ciò si realizza tipicamente tramite impostazioni demandate a comandi `\pgfplotsset`. Esse saranno inserite nel preambolo del singolo sorgente dedicato allo specifico grafico (volta per volta o caricate da un file fisso tramite la macro `\input`).

Nel seguente listato è riportato un *template* di sorgente basato sulla classe standalone:

```
% preambolo
\documentclass{standalone}
\usepackage{lmodern}
\usepackage{pgfplots}
\input{% comandi \pgfkeys e \pgfplotsset
   ⟨file di configurazione⟩}
...
% corpo documento
\begin{document}
\begin{tikzpicture}
   \begin{axis}[⟨opzioni grafiche⟩]
      ...
      ⟨comandi di pgfplots o di tikz⟩
      ...
   \end{axis}
\end{tikzpicture}
\end{document}
```

Con questo accorgimento si ottiene una dimensione finale della pagina tale da contenere tutti gli elementi grafici senza inutili contorni bianchi. Il risultato è praticamente simile a quello ottenibile scontornare una pagina con `pdfcrop`.

4. Il codice che genera la figura 5 è reperibile all'indirizzo: `http://tex.stackexchange.com/questions/25775/fill-area-of-curve-above-a-line-using-pgfplots`.

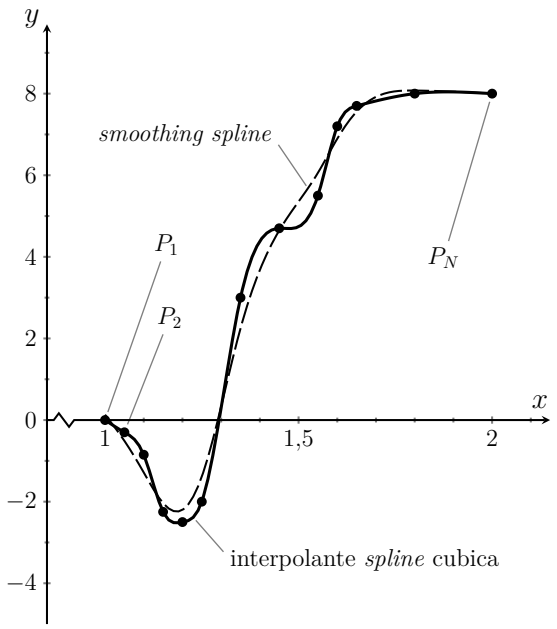


FIGURA 6: Grafico di funzioni interpolanti. I punti di supporto (pallini) sono i medesimi della figura 4.

Un esempio di curve interpolanti costruite a partire dai dati della figura 4a è mostrato nella figura 6. Se, ad esempio, quei dati rappresentano grandezze fisiche potrebbe essere lecito assumere che esista una legge quantitativa che lega l'insieme dei numeri x_i a quello dei numeri y_i. Detta $y_i = f_{\text{tab}}(x_i)$ la funzione tabellare, spesso si vuole trovare una opportuna funzione $g(x)$ che *interpola* gli N valori y_i. La funzione interpolante dovrebbe descrivere sufficientemente bene il fenomeno o il contesto che determina le coppie $(x_j.y_j)$, consentendo di fare delle previsioni sul valore della variabile dipendente y per valori di x diversi dagli x_i. Nel contesto della Teoria dell'interpolazione le coppie $(x_i.y_i)$ prendono il nome di *punti di supporto* e le ascisse di supporto x_i vengono chiamate anche *nodi*. Spesso, quando si vuole tracciare una curva a partire dall'insieme delle (x_i, y_i) non si vuole fare altro che disegnare il grafico di una funzione interpolante $g(x)$. La spezzata della figura 4b è un esempio di grafico di una funzione interpolante lineare a tratti.

Oggi esistono ambienti di lavoro sofisticati come Matlab, Octave, Mathematica o Mathcad tramite i quali l'utente ha già la possibilità di produrre grafici. Queste applicazioni, tra le tante caratteristiche che le distinguono, sono adatte alla ricerca di funzioni interpolanti. D'altra parte, quando si presenta l'esigenza di ottenere un grafico di buona qualità tipografica. è naturale ricorrere a PGFPLOTS. L'utente esporterà i dati in un formato opportuno per poterli usare nell'ambiente axis attraverso il comando `\addplot`.

La curva riportata nella figura 6 in linea con-

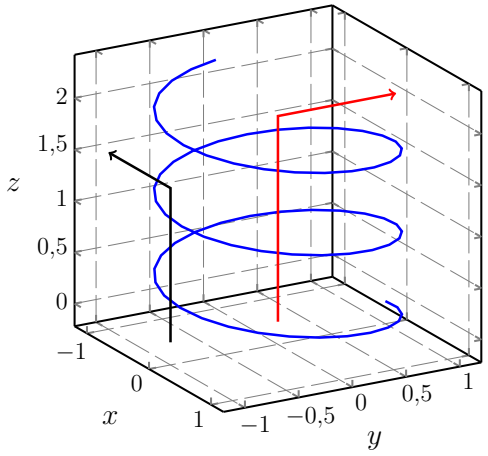


FIGURA 7: Un'elica disegnata con `\addplot3`.

tinua è il grafico di un'interpolante *spline* cubica ricavata in Matlab. Essa è detta interpolante *esatta* nel senso che per ogni x_i è $y_i = g(x_i)$. Nella stessa figura è mostrata anche una curva interpolante *approssimata* detta *smoothing spline* (curva tratteggiata). Essa non passa per i punti di supporto e la sua "morbidezza" è regolata secondo un determinato criterio. Le curve interpolanti sono pur sempre tracciate unendo dei segmenti, ma in questo caso il numero di punti utilizzati è pari a $5N$ e non si ha la percezione di osservare delle spezzate. Le coordinate sono state calcolate in Matlab ed esportate in un file di testo. Successivamente sono state caricate tramite il comando `\addplot` dando la direttiva `file`.

Il manuale d'uso di PGFPLOTS fa anche esplicito riferimento al programma `matlab2tikz.m`[5] messo a disposizione degli utenti Matlab. Questo *script* permette di tradurre gli elementi di un grafico ottenuto con Matlab in un ambiente axis con opportune impostazioni, che racchiude gli appositi comandi `\addplot`. È anche possibile produrre in un solo colpo un sorgente LaTeX la cui compilazione produce direttamente il file PDF con il grafico.

6 Creare grafici tridimensionali

Con PGFPLOTS esiste la possibilità di rappresentare curve e superfici dello spazio tridimensionale. Per quanto riguarda il tracciamento di curve esiste il comando `\addplot3`. Le modalità d'uso sono simili a quelle di `\addplot` con la differenza che `\addplot3` lavora su terne di coordinate.

La curva a forma di elica nella figura 7 si ottiene definendo tre espressioni matematiche, una per ciascuna coordinata, funzioni di un parametro reale. Il disegno è prodotto con un sorgente che assomiglia a quello seguente:

```
\begin{tikzpicture}
\begin{axis}[
   view={60}{20},      % punto di vista
   xlabel=$x$, ylabel=$y$, zlabel=$z$,
   variable=\t         % il nome del parametro
   ]
% L'elica
\addplot3 +[ % impostazioni aggiuntive
   domain=0:5.5*pi,
   samples=70,
   samples y=0, % per non creare inutili griglie
   no marks
   ]                   % curva parametrica
   (sin(deg(t)),       % x(t)
      cos(deg(t)),     % y(t)
         2*t/(5*pi)); % z(t)
% la spezzata
\addplot3 +[->,no marks] coordinates {
   (0,0,0) (0,0,2) (0,1.1,2) };
% comando di tikz
\draw[->] (axis cs:0,-1,0) % terne di coordinate
   -- (axis cs:0,-1,1.5) -- (axis cs:-1,-1,1.5);
\end{axis}
\end{tikzpicture}
```

Anche il comando `\addplot3` accetta le direttive `coordinates` e `file`. Relativamente al tracciamento di curve il funzionamento è analogo a quanto già visto per il comando `\addplot`. Il codice precedente mostra anche due modi differenti con cui disegnare una spezzata che congiunge terne di coordinate successive: nel primo caso attraverso l'uso stesso di `\addplot3` dando la direttiva `coordinates`, nel secondo caso tramite il comando `\draw` di Ti*k*Z ed utilizzando il sistema di riferimento `axis cs`.

Anche la rappresentazione di una superficie avviene tramite `\addplot3` specificando l'opzione surf. Si rimanda al manuale d'uso di PGFPLOTS per approfondimenti ed esempi sul modo in cui vanno trattati i dati da fornire in pasto al comando di disegno. A titolo di esempio riportiamo nella figura 8 una superficie generata in Matlab e rappresentata con l'ausilio di `matlab2tikz.m`.

7 Esempi ragionati

In questa sezione abbiamo ritenuto utile illustrare alcuni esempi ragionati. Lo scopo è di continuare a mostrare nuovi dettagli sulle numerose possibilità di rappresentazione offerte da PGFPLOTS, con la

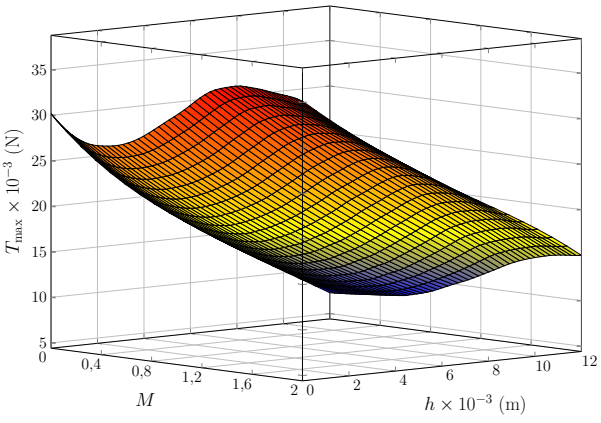


FIGURA 8: Superficie prodotta con l'ausilio di `matlab2tikz.m`.

5. Si veda `https://github.com/nschloe/matlab2tikz`

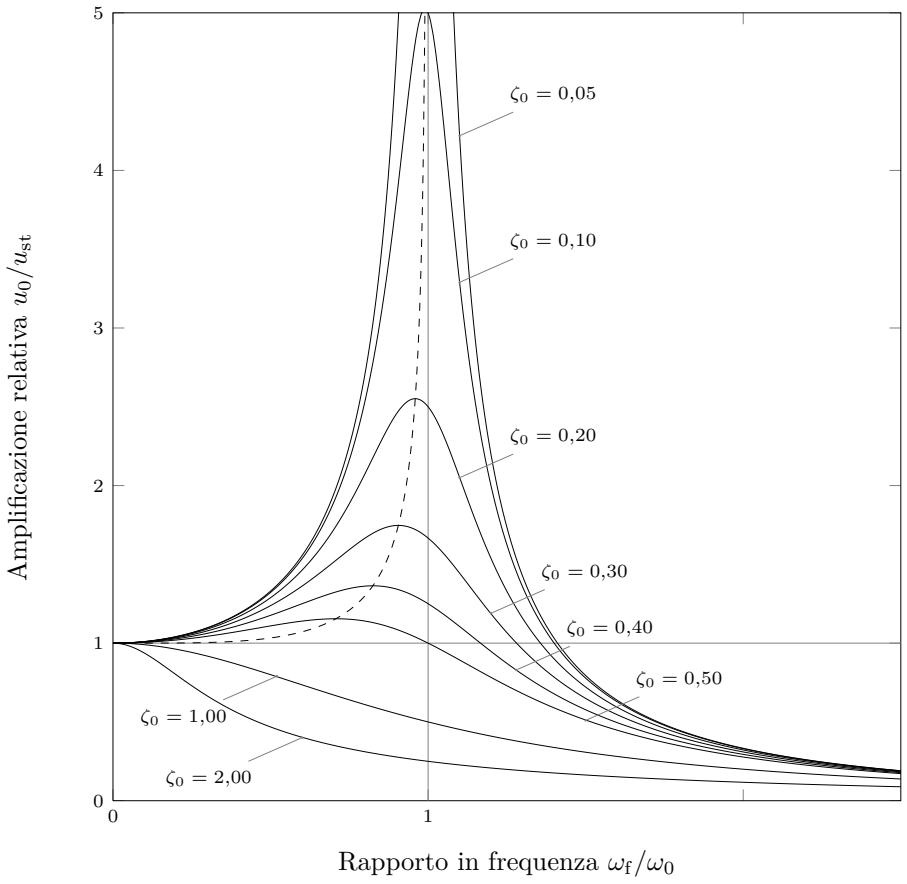


FIGURA 9: La famiglia di curve di risonanza dell'oscillatore semplice.

speranza che quanto detto finora abbia invogliato il lettore a provare il pacchetto per le proprie necessità.

7.1 La risonanza

Vediamo adesso come se la cava PGFPLOTS se vogliamo mostrare una famiglia di curve sullo stesso grafico. Ci rivolgiamo ad un problema classico della dinamica tecnica: la *risonanza* in un oscillatore semplice soggetto ad una forza periodica di eccitazione, problema che reca non pochi grattacapi agli ingegneri antisismici.

Il coefficiente di amplificazione varia molto repentinamente quando la frequenza dell'eccitazione ω_f si avvicina alla pulsazione propria del sistema ω_0, appunto per il fenomeno della risonanza, quindi decidiamo senza indugio di prevenire i problemi di precisione numerica di TEX rivolgendoci al programma esterno gnuplot il cui uso è perfettamente integrato in PGFPLOTS. Ne risulta il grafico riportato nella figura 9 si caratterizza per una serie di particolarità grafiche.

7.1.1 Famiglia di curve

Innanzi tutto vorremmo disegnare una famiglia di curve con lo stesso stile di linea, ma ad ogni comando `\addplot` lo stile verrebbe scelto ciclicamente su una sequenza di default. Se non si vuole assegnare lo stesso stile ad ogni curva è sufficiente configurare PGFPLOTS per utilizzare uno stile ciclico che contiene un unico stile per linea continua in colore nero con l'opzione cycle list.

L'opzione line width dell'ambiente axis regola lo spessore del tratto in cui viene disegnato il riquadro della tela ma anche il tratto delle curve delle funzioni. Rimane sempre la possibilità di regolare questa proprietà tra le opzioni del comando `\addplot`, come si è fatto nel caso in esame per disegnare la curva dei massimi con stile di linea tratteggiato.

7.1.2 Ottimizzazione del grafico

In corrispondenza del valore 1 le curve di risonanza assumono valori via via crescenti, ma il pacchetto PGFPLOTS *taglia* il grafico correttamente applicando i valori degli intervalli di dominio. Serve solo preoccuparsi di regolare la definizione delle coordinate della tela con le coppie di opzioni xmin, xmax e ymin, ymax, per aggiustare la *finestra* del grafico con lo scopo di ottenere il miglior risultato.

Per rifinire la rappresentazione non rimane che utilizzare normali comandi PGF per aggiungere le due linee di riferimento orizzontale e verticale nell'area del grafico con l'uso di coordinate direttamente espresse con riferimento agli assi del grafico nel sistema axis cs, regolare le etichette degli assi stessi su valori particolari e non regolari (opzioni

xtick e ytick per le posizioni coordinate e xticklabels per il testo d'etichetta da assegnare alle marche).

Finora tutte le esigenze sono state brillantemente risolte da PGFPLOTS, soprattutto quella fondamentale di consentire all'utente di regolare facilmente dimensioni ed intervalli di definizione allo scopo di raggiungere la rappresentazione più significativa del problema della risonanza. Nel listato seguente è riportato il codice completo del grafico:

```
\documentclass{standalone}
\usepackage{lmodern,pgfplots}
\begin{document}
\tikzset{
  every pin/.style={font=\scriptsize,pin
     distance=4ex},
  small dot/.style={fill=gray,circle,scale=0.1}
}
\begin{tikzpicture}
\begin{axis}[
    tick label style={font=\scriptsize},
    xlabel={Rapporto in frequenza
            $\omega_\mathrm{f}/\omega_0$},
    ylabel={Amplificazione relativa
            $u_0/u_\mathrm{st}$},
    ytick={0,1,2,3,4,5},
    xtick={0,1,2,3},
    xticklabels={0,1,,},
    %
    % opzioni di disegno globali
    no markers,
    line width=0.3pt,
    cycle list={{black,solid}},
    %
    % dominio 2D
    samples=200,
    smooth,
    domain=0:2.5,
    xmin=0, xmax=2.5,
    ymin=0, ymax=5.0,
    %
    % dimensioni della tela
    width=12cm,
    height=12cm
    ]
% horizontal help line
\draw[help lines] (axis cs:0,1) -- (axis cs:2.5,1);
% vertical help line
\draw[help lines] (axis cs:1,0) -- (axis cs:1,5);
% tracciamento delle curve
\addplot gnuplot {1/sqrt((1-x^2)^2+4*0.05^2*x^2)};
\addplot gnuplot {1/sqrt((1-x^2)^2+4*0.10^2*x^2)};
\addplot gnuplot {1/sqrt((1-x^2)^2+4*0.20^2*x^2)};
\addplot gnuplot {1/sqrt((1-x^2)^2+4*0.30^2*x^2)};
\addplot gnuplot {1/sqrt((1-x^2)^2+4*0.40^2*x^2)};
\addplot gnuplot {1/sqrt((1-x^2)^2+4*0.50^2*x^2)};
\addplot gnuplot {1/sqrt((1-x^2)^2+4*1.00^2*x^2)};
\addplot gnuplot {1/sqrt((1-x^2)^2+4*2.00^2*x^2)};
% tracciamento della curva dei massimi
\addplot gnuplot[dashed,domain=0:0.99]
   {1/sqrt(1-x^4)};
% etichette delle curve
\node[small dot,pin=30:{$\zeta_0=0{,}05$}] at
   (axis cs:1.10,4.22) {};
\node[small dot,pin=30:{$\zeta_0=0{,}10$}] at
   (axis cs:1.10,3.29) {};
\node[small dot,pin=30:{$\zeta_0=0{,}20$}] at
   (axis cs:1.10,2.05) {};
\node[small dot,pin=30:{$\zeta_0=0{,}30$}] at
   (axis cs:1.20,1.19) {};
\node[small dot,pin=30:{$\zeta_0=0{,}40$}] at
   (axis cs:1.28,0.83) {};
\node[small dot,pin=30:{$\zeta_0=0{,}50$}] at
   (axis cs:1.50,0.51) {};
\node[small dot,pin=210:{$\zeta_0=1{,}00$}] at
   (axis cs:0.52,0.79) {};
\node[small dot,pin=210:{$\zeta_0=2{,}00$}] at
   (axis cs:0.60,0.40) {};
\end{axis}
\end{tikzpicture}
\end{document}
```

7.1.3 Il problema delle etichette

L'etichetta che individua ciascuna curva può essere apposta con l'elegante oggetto grafico `pin` del pacchetto PGF, che consiste in una linea che unisce un testo, composto anche in modo matematico, con un punto del grafico. Occorre fare in modo che questo punto appartenga alla curva corrispondente.

Tuttavia l'oggetto `pin` non sa niente della curva che etichetta. Ne consegue che occorre un lungo lavoro di raffinamento della posizione dei punti sulle curve. Ciò è normale se pensiamo che l'etichetta va posta senza sovrapposizioni ed in modo tale che non ci sia alcun dubbio su quale curva sia stata identificata. Tuttavia ricalcolare il valore della funzione ogni volta che si desidera mutare l'ascissa del punto dell'oggetto `pin`, non è né comodo né elegante.

Come si vedrà nel prossimo paragrafo, questo inconveniente può essere superato brillantemente facendo ricorso alle funzionalità avanzate messe a disposizione dal pacchetto tikz, creando un nuovo componente grafico che interagisce con la curva da etichettare.

Il problema dell'etichettatura si riduce essenzialmente nel conoscere il punto di aggancio dell'oggetto `pin`. Dal punto di vista numerico, il modo per risolverlo è far calcolare le coordinate automaticamente. Sfortunatamente, per problemi di espansione, non è possibile inserire l'espressione matematica direttamente nelle coordinate del punto di aggancio.

L'uso della libreria PGFmath è comunque possibile scrivendo un comando poco elegante che memorizza il valore numerico dell'ordinata ogni volta in una macro diversa per poi farla espandere nelle coordinate del nodo:

```
\def\eval#1#2#3{% #1 -> nome macro
                % #2 -> ascissa
                % #3 -> smorzamento
   \pgfmathparse{1/sqrt((1-#2^2)^2+4*#3^2*#2^2)}
   \expandafter\edef\csname #1\endcsname{%
                        \pgfmathresult}
}
...
% etichetta curva (in ambiente axis)
\eval{a}{1.15}{0.05}
\node at (axis cs:1.15,\a) % <- ordinata
   [small dot,
    pin={30:{$\zeta_0=0{,}05$}}] {};
...
```

Idea alternativa appena un po' più efficace è quella di usare la libreria matematica di Lua disponibile all'interno di LuaLaTeX. Nel seguente frammento di codice la macro `\eval` viene espansa direttamente nel valore numerico dell'ordinata con 4 cifre significative grazie alla nuova primitiva `\directlua` di LuaTeX[6].

```
\def\eval#1#2{% #1-> ascissa, #2->smorzamento
   \directlua{
      local v = 1/math.sqrt((1-#1^2)^2 +
                4*#2^2*#1^2)
      tex.sprint(string.format("\%0.4f",v))
   }
}
...                                % (in ambiente axis)
% etichetta curva
\node at (axis cs:1.15,\eval{1.15}{0.05})
   [small dot,
    pin={30:{$\zeta_0=0{,}05$}}] {};
...
```

Il linguaggio Lua (Ierusalimschy, 2006) estende notevolmente le capacità elaborative di TeX. Nell'esempio, il codice Lua all'interno della macro `\directlua` prima di essere eseguito viene *espanso* così che gli argomenti in stile TeX ed il carattere di percentuale costruiranno codice Lua corretto. La funzione `tex.sprint()` inserirà poi il dato numerico nel normale flusso di gestione dei token.

L'esempio minimo dimostra come in LuaTeX si possono facilmente scambiare dati *da* e *verso* Lua. Si spera che tutta questa potenza elaborativa venga ben impiegata nelle nuove generazioni di componenti software per i sistemi TeX, a tutto beneficio degli utenti.

7.2 Etichettatura semiautomatica senza impiegare strumenti esterni

Riprendiamo il problema delle etichette affrontato nell'esempio precedente e vediamo una soluzione alternativa basata esclusivamente sulle funzionalità offerte da PGF e PGFPLOTS.

Riassumendo, quando si vuole disegnare un insieme di più curve nello stesso sistema di assi cartesiani è possibile facilitare la lettura del grafico in due modi: inserire una legenda corredata di segni grafici che rimandano a ciascuna curva oppure etichettare direttamente le singole curve con delle marcature opportune. Quando si sceglie la seconda di queste due tecniche — spesso perché c'è una particolare eigenza di chiarezza — è possibile evitare di apporre una marcatura "per tentativi"[7]. Andiamo a impostare il lavoro di configurazione iniziale, per andare poi a realizzare il disegno mostrato nella figura 10.

Il procedimento che illustreremo di seguito si basa sulla possibilità offerta dal pacchetto tikz di definire degli stili personalizzati con il comando `\pgfkeys`. Questa tecnica richiede di caricare la libreria intersections. Nel preambolo del documento si daranno le due istruzioni seguenti:

```
% nel preambolo
\usepackage{pgfplots}
\usetikzlibrary{intersections}
```

A questo punto si andrà a definire un nuovo `tikzstyle` che chiameremo `linelabel`. Esso sarà a disposizione dell'utente sotto forma di opzione del comando `\addplot`.

Il nuovo stile crea una linea verticale (un tracciato nascosto, *path*), staccata in corrispondenza di una certa ascissa; insieme alla linea verticale viene creato un punto d'intersezione con la curva da etichettare. La comodità di questo procedimento, detto $[a, b]$ l'intervallo delle x sotteso dalla curva, risiede nella possibilità da parte dell'utente di specificare la posizione adimensionale $\xi = (x-a)/(b-a)$ della linea verticale. Ciò è realizzato attraverso il seguente frammento di codice:

```
\pgfkeys{
   /pgfplots/linelabel/.style args={#1:#2:#3}{%
     name path global=labelpath,
     execute at end plot={
        \path [name path global=labelpositionline]
            (rel axis cs:#1,0) --
                (rel axis cs:#1,1);
        \draw [help lines,text=black,
            inner sep=0pt,
            name intersections={
                of=labelpath and labelpositionline
            }]
            (intersection-1) -- +(#2)
                node [label={#3}] {};
     }
   }
}
```

L'opzione linelabel accetta tre argomenti. Il primo argomento è il valore della ξ (0 per l'estremità sinistra, 1 per l'estremità destra). Il secondo argomento è la posizione dell'etichetta rispetto al punto d'intersezione della linea verticale con la curva da etichettare; tale posizione è assegnabile da una coppia di coordinate polari ($\Delta\theta$,$\Delta\rho$) o cartesiane (Δx,Δy) separate da una virgola. Infine, il terzo argomento è costituito dal codice dell'etichetta, ovvero un testo con eventuali opzioni di decorazione e posizionamento.

Così, ad esempio, la curva corrispondente alla funzione x^2 nella figura 10 verrà disegnata dal seguente codice:

```
\addplot [linelabel=
          0.8:                          % --> ξ
          {135:1.75cm}:                 % --> Δθ, Δρ
         {[black]above left:$x^2$} % etichetta
   ] {x^2};
```

Si osservi che i tre argomenti dello stile `linelabel` devono essere separati dal carattere due punti (`:`).

Un esempio più raffinato è mostrato nella stessa figura 10 ed è dato dalla curva disegnata con tratto

6. Ad oggi il nuovo motore di composizione LuaTeX non ha ancora raggiunto una versione stabile ed è disponibile per esempio installando una distribuzione TeX Live.

7. Si veda la discussione: `http://tex.stackexchange.com/questions/12207/label-plots-in-pgfplots-without-entering-coordinates-manually`

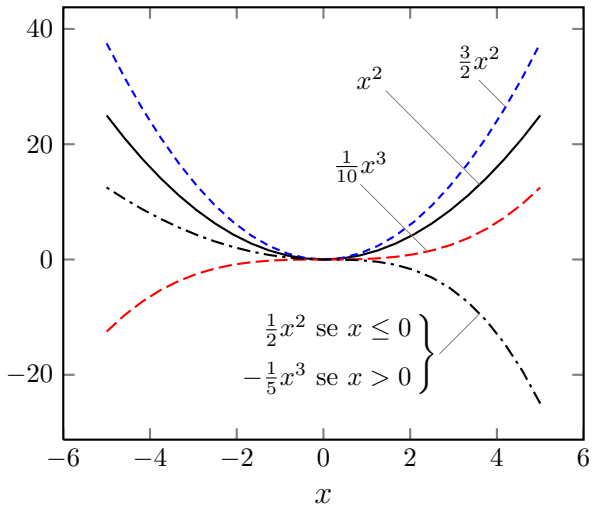


FIGURA 10: Curve multiple etichettate direttamente attraverso la definizione di un nuovo `tikzstyle`.

e punto. Essa è il grafico di una funzione definita come $x^2/2$ per $x < 0$ e come $-x^3/5$ per $x \ge 0$. Il codice che disegna la curva è il seguente:

```
\addplot [thick,
    dash pattern=on 1.2pt off 2pt on 5pt off 2pt,
    linelabel=
    0.80:                       % ⟶ ξ
    {-135:0.75cm}:              % ⟶ Δθ, Δρ
    {left:                      % etichetta
      $
        \begin{array}{rl}
          \frac{1}{2}x^2 &\text{se } x < 0\\
          -\frac{1}{5}x^3 &\text{se } x \ge 0
        \end{array}\biggr\}
      $
    }
  ] {(x<0)*0.5*x^2 + (!(x<0))*(-0.20*x^3)};
```

L'ultima riga di codice mostra anche la sintassi necessaria a definire funzioni in maniera differente in differenti intervalli.

Un secondo esempio in cui è stata applicata la tecnica di etichettatura su illustrata è dato dalla figura 11. In questo caso, in cui non è stato preferito l'uso di una legenda delle curve, si vede chiaramente che è necessario utilizzare una marcatura con `pin`.

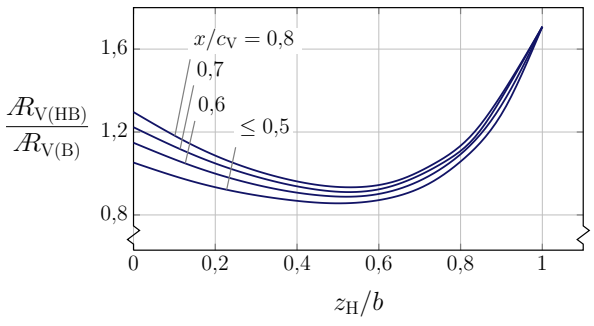


FIGURA 11: Un altro esempio di grafico in cui l'etichettatura delle curve è stata ottenuta in maniera semiautomatica.

7.3 Regolare la marcatura degli assi

In questo esempio vogliamo dare i dettagli sulle opzioni necessarie a disegnare i grafici della figura 12. Cominciamo con la figura 12c. Le impostazioni più significative sono contenute nel seguente frammento di codice:

```
% nel preambolo
\usepackage{siunitx}
\sisetup{
   unitsep=thin, valuesep=thin,
   decimalsymbol=comma,
   expproduct=cdot, digitsep=thin,
   sepfour=false, group-digits=false
}
\newunit{
   \kilopond}{kg\ensuremath{_\mathrm{f}}
}
% ...
\begin{document}
\begin{tikzpicture}
\begin{axis}[
   % ...
   xmin=0, xmax=2.0,
   xtick={0,0.4,...,2.4},
   ymin=500, ymax=4000,
   ytick={500,1000,...,4000},
   xlabel={$M$},
   ylabel={
     \parbox{2em}{
        \centering$T_\mathrm{max}$\\
        \centering(\SI{}{\kilopond})}
   }
   ]
% tracciatura delle curve
\addplot[  ] coordinates {
   ...
   };
% ...
% etichettatura delle curve
% ...
\end{tikzpicture}
```

Si riconoscono i comandi caricano il pacchetto siunitx e definiscono l'unità di misura `\kilopond` (non appartenente al Sistema Internazionale). Essa viene utilizzata per comporre l'etichetta dell'asse delle ascisse.

Ciò rende interessante questo esempio è l'aspetto delle *tick label* dell'asse delle ordinate. Per come è impostata la scala delle ordinate le etichette di marcatura appaiono troppo ingombranti ("500", "1000", fino a "4000"). In questi casi viene in aiuto la possibilità di scalare i valori numerici delle etichette attraverso delle opzioni opportune dell'ambiente axis. Il dettaglio di un grafico simile a quello della figura 12c è mostrato nella figura 12b; la differenza è che in quest'ultima le etichette di marcatura sono più compatte avendo apposto il segno "$\cdot 10^3$" in cima all'asse delle ordinate. Ciò è possibile grazie all'opzione scaled y ticks e allo stile `ytick scale label code`. Il frammento di codice seguente illustra il loro uso:

```
% ...
\begin{axis}[
   % ...
```

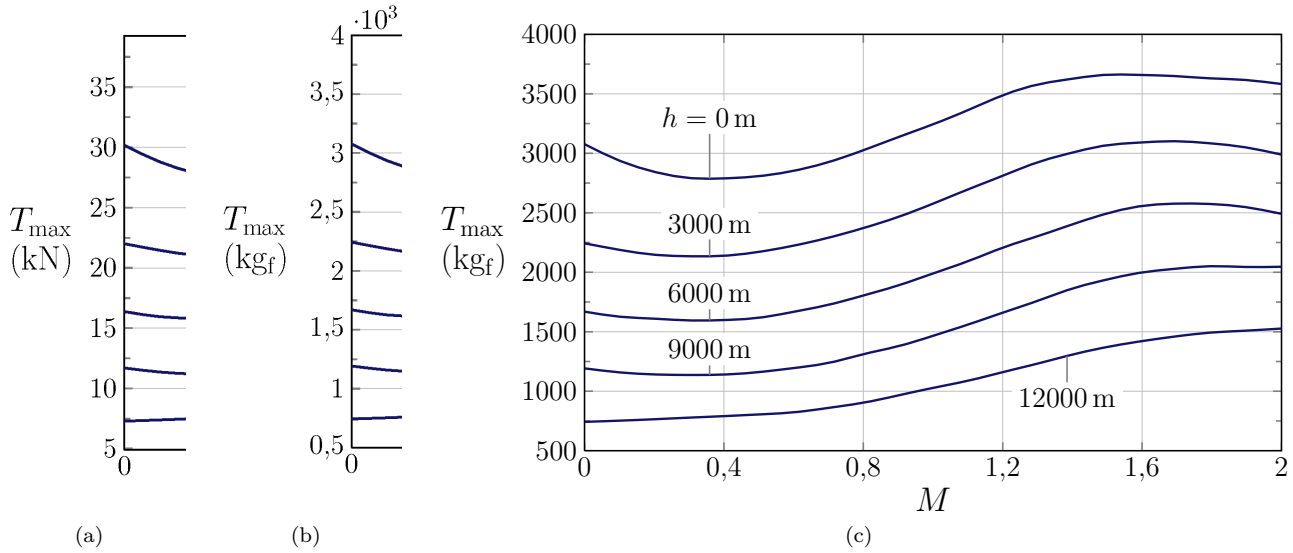


FIGURA 12: Regolare i valori delle etichette numeriche.

```
    xmin=0, xmax=2.0,
    xtick={0,0.4,...,2.4},
    ymin=500, ymax=4000,
    ytick={500,1000,...,4000},
    % divide per 1000 le etichette
    scaled y ticks=real:1000,
    % fattore di scala "·10^3" in alto
    ytick scale label code/.code={$\cdot 10^3$},
    % da qui il codice è uguale a quello precedente
    % ...
]
% ...
\end{axis}
```

Si noti che il ritocco delle etichette marcatura proposto sopra è avvenuto senza modificare i dati e i comandi di tracciamento delle curve.

Per finire, si esamini la figura 12a. I dati sono disponibili in kilopond, un'unità di misura del cosiddetto Sistema Tecnico, ma per aderire alle norme unificate sarebbe opportuno riportare il Newton (N) o un suo multiplo, l'unità di misura del Sistema Internazionale. Ciò può essere fatto ancora una volta senza elaborare i dati ma scalando opportunamente sia le etichette di marcatura che i valori numerici che definiscono i *tick mark* (opzione ytick). Il frammento di codi seguente illustra la soluzione proposta:

```
\begin{axis}[
    xmin=0, xmax=2.0,
    xtick={0,0.4,...,2.4},
    ymin=500, ymax=4000,
    ytick={%
                509.683996, %   5000/9.81
               1019.36799,  %  10000/9.81
               ... ,
               4077.47197   %  40000/9.81
    },
    % divide per (1000/9.81)
    scaled y ticks=real:101.936799184506,
    ytick scale label code/.code={}, % vedi y label
    xlabel={$M$},
    ylabel={
        \parbox{2em}{
            \centering$T_\mathrm{max}$\\
            \centering(\SI{}{\kilo\newton})}
    }
]
% ...
```

7.4 Spettri a colori

Una delle visualizzazioni classiche che compaiono nelle relazioni di calcolo dei progetti in zona sismica è il grafico degli *spettri di risposta*. Con il prossimo esempio dimostreremo come sia possibile con una manciata di righe di codice, costruire gli spettri con PGFPLOTS assegnando al comando `\addplot`, direttamente le espressioni matematiche riportate nella nuova normativa italiana (il D.M. 14/01/2008).

Uno spettro sismico è una curva composta da quattro funzioni ciascuna con il proprio intervallo di dominio. Nell'ambiente axis daremo quindi altrettanti comandi `\addplot` consecutivi specificando per ciascuno di essi l'intervallo di validità con l'opzione essenziale domain. Vorremo anche riferirci nel codice del grafico alle grandezze in gioco con il proprio significato fisico, quindi esprimeremo le curve in funzione del *periodo* T, cambiando la variabile con la direttiva variable=\⟨*var*⟩ (notare come la nuova variabile deve essere preceduta obbligatoriamente con il carattere di escape backslash), mentre introdurremo i valori dei dati con macro standard di TEX:

```
\documentclass{standalone}
\usepackage{lmodern}
\usepackage{pgfplots}

\begin{document}
\begin{tikzpicture}
% defining personal drawing style
\pgfplotsset{
  slv/.style={line width=1pt,color=blue},
```

```
}

\begin{axis}[
    % etichette del grafico
      tick label style={font=\scriptsize},
      xlabel={$T$ (s)},
      ylabel={$S_e/g$},
      xtick={0,0.5,...,3.0},
      grid=major,
    % linee di plottaggio
      no markers,
    % dominio 2D
      smooth,
      xmin=0, xmax=3.00,
      ymin=0, ymax=0.55,
    % dimensioni tela
      width=13cm,
      height=8cm,
    % variabile nelle espressioni: periodo T
      variable=\T
     ]

\def\ag{0.15182}   % accelerazione
\def\Fo{2.402}     % coefficiente Fo
\def\Ss{1.10}      % effetto locale
\def\Tb{0.096}     % periodo punto B
\def\Tc{0.289}     % periodo punto C
\def\Td{2.207}     % periodo punto D

\addplot[slv,domain=0.0:\Tb]
    {\ag*\Ss*\Fo*(T/\Tb+(1/\Fo)*(1-T/\Tb))};
\addplot[slv,domain=\Tb:\Tc]
    {\ag*\Ss*\Fo};
\addplot[slv,domain=\Tc:\Td]
    {\ag*\Ss*\Fo*\Tc/T};
\addplot[slv,domain=\Td:5.0]
    {\ag*\Ss*\Fo*\Tc*\Td/T^2};

% legenda
\legend{\textsc{slv},,,}
\end{axis}
\end{tikzpicture}
\end{document}
```

Molto comodo per definire gli intervalli di marcatura sull'asse delle ascisse è utilizzare la notazione *foreach*, nota in PGF, dove i valori intermedi sono sostituiti dai tre puntini digitati fra i primi due valori che fissano inizio ed intervallo della serie, ed il valore finale, mentre la legenda viene creata inserendo argomenti *vuoti* con una sequenza di caratteri virgola corrispondentemente ai successivi rami dello spettro dopo il primo.

Il linguaggio del codice esprime in modo semplice e diretto sia le caratteristiche ingegneristiche del problema rappresentato, sia le proprietà visuali del grafico, ma si può ancora migliorare l'output notando che i vari rami dello spettro non si raccordano agli estremi non essendo un *path* continuo. Il ritocco può essere risolto semplicemente disegnando nei punti di discontinuità della curva un circoletto pieno del diametro e colore corrispondente a quello della linea di plottaggio dello spettro. Per realizzare effettivamente questo trucco basta inserire nell'ambiente axis una normale istruzione di disegno PGF con la posizione del punto della discontinuità, ed in questo ci viene in aiuto PGFPLOTS mettendoci a disposizione i nodi relativi ai punti di inizio e fine curva `current plot begin` e `current plot end`. Ecco come potrebbe essere il comando che disegna il circoletto dopo aver plottato il primo ramo:

```
\fill (current plot end) circle [radius=0.5pt];
```

L'ultima osservazione è che il secondo ramo dello spettro è un semplice segmento orizzontale potrebbe quindi essere disegnato inserendone i punti di estremità senza considerarlo come fatto finora in termini di funzione dipendente. Ebbene il comando `\addplot` prevede come opzione l'inserimento di un *path* in coda all'argomento principale chiamato ⟨*trailing path commands*⟩, basterà quindi aggiungere il path di un segmento come avremmo fatto per un normale `\draw` di PGF subito dopo l'espressione matematica tra parentesi graffe. Il tratto orizzontale inizia dal punto `(current plot end)` e termina nel punto `(axis cs:\Tc,\ag*\Ss*\Fo)`. Purtroppo nel sistema `axis cs` le coordinate riferite agli assi coordinati non vengono elaborate come espressioni complesse come quella dell'ordinata dell'esempio ma esclusivamente come numeri. Dovremo quindi scrivere il secondo estremo come `(axis cs:\Tc,0.40114)` eseguendo manualmente la moltiplicazione dei tre valori oppure potremo rinunciare alle coordinate riferite al grafico usando quelle assolute (in cui è possibile inserire operazioni di calcolo), sapendo come *dirty tricks* che quelle locali lineari si ottengono moltiplicando per 100 il valore di ascissa e per 1000 il valore dell'ordinata, come si è fatto nel seguente frammento di codice che disegna i primi due rami dello spettro:

```
\addplot[slv,domain=0.0:\Tb]
    {\ag*\Ss*\Fo*(T/\Tb+(1/\Fo)*(1-T/\Tb))}
    (current plot end)--(100*\Tc,1000*\ag*\Ss*\Fo);
```

A questo punto non rimane che definire una macro `\spettro` che attende il colore ed i parametri del sito per plottare sullo stesso grafico quattro spettri standard di progetto come illustra il codice seguente con il quale si aggiungono nell'area del grafico le informazioni testuali di base utilizzate per la costruzione degli spettri. Il risultato è mostrato in figura 13.

```
\documentclass{standalone}
\usepackage{lmodern}
\usepackage{pgfplots}
\usepackage[arc-separator= \,]{siunitx}

\begin{document}
\begin{tikzpicture}
% defining personal drawing style
\pgfplotsset{sl/.style={line width=1pt}}

\begin{axis}[
    % etichette del grafico
      tick label style={font=\scriptsize},
      xlabel={$T$ (s)},
      ylabel={$S_e/g$},
      xtick={0,0.5,...,3.0},
      grid=major,
```

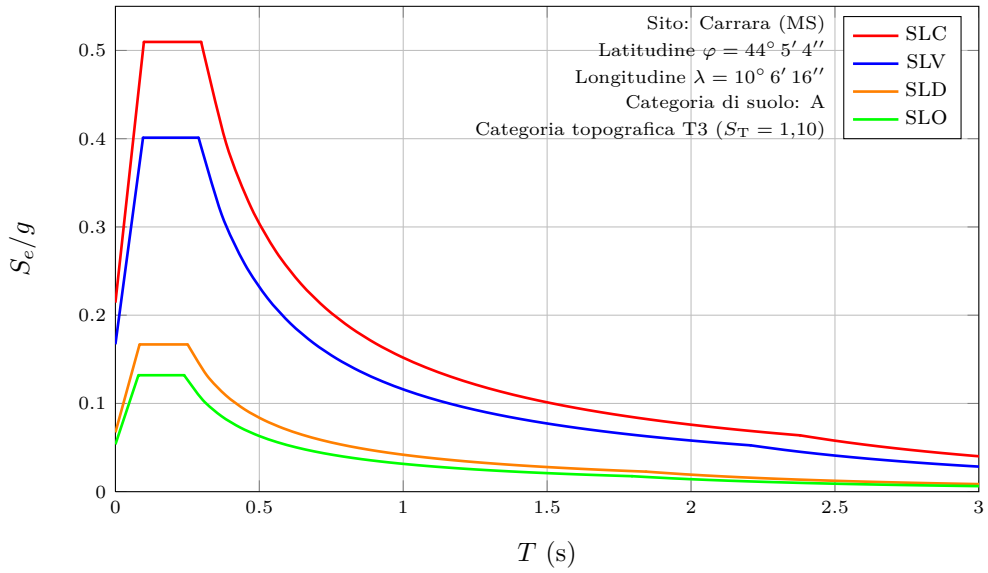


FIGURA 13: Spettri di risposta come curve a più rami

```
    % linee di plottaggio
      no markers,
    % dominio 2D
      smooth,
      xmin=0, xmax=3.00,
      ymin=0, ymax=0.55,
    % dimensioni tela
      width=13cm,
      height=8cm,
    % variabile nelle espressioni: periodo T
      variable=\T
     ]

\newcommand\spettro[6]{
\addplot[color=#1,sl,domain=0.0:#4]
     {#2*1.10*#3*(T/#4+(1/#3)*(1-T/#4))}
     (current plot end)--(100*#5,1000*#2*1.10*#3);

\fill[color=#1] (current plot end)
                circle [radius=0.5pt];
\addplot[color=#1,sl,domain=#5:#6]
                {#2*1.10*#3*#5/T};
\fill[color=#1] (current plot begin)
                circle [radius=0.5pt];
\addplot[color=#1,sl,domain=#6:5.0]
                {#2*1.10*#3*#5*#6/T^2};
\fill[color=#1] (current plot begin)
                circle [radius=0.5pt];
}

% slc, slv, sld, slo
\spettro{red}{0.1947}{2.38}{0.099}{0.298}{2.379}
\spettro{blue}{0.1518}{2.40}{0.096}{0.289}{2.207}
\spettro{orange}{0.061}{2.48}{0.084}{0.251}{1.844}
\spettro{green}{0.0487}{2.46}{0.080}{0.239}{1.795}

% legenda
\legend{\textsc{slc},,,
        \textsc{slv},,,
        \textsc{sld},,,
        \textsc{slo},,,}

% add text info to the graph
\draw (axis cs:2.5,0.53) node[left]
{\scriptsize Sito: Carrara (MS)};
\draw (axis cs:2.5,0.50) node[left]
{\scriptsize Latitudine $\varphi=\ang{44;5;4}$};
\draw (axis cs:2.5,0.47) node[left]
{\scriptsize Longitudine $\lambda=\ang{10;6;16}$};
\draw (axis cs:2.5,0.44) node[left]
{\scriptsize Categoria di suolo: A};
\draw (axis cs:2.5,0.41) node[left]
{\scriptsize
Categoria topografica T3 ($S_\mathrm{T}=1{,}10$)};
\end{axis}
\end{tikzpicture}
\end{document}
```

7.5 Piano semilogaritmico

Come esempio di piano semilogaritmico tracciamo le curve di probabilità poissoniana in funzione della frequenza media di accadimento di un evento. Questa volta l'ambiente da usare sarà quindi semilogxaxis.

Nel codice che genera il grafico di figura 14 riportato per intero di seguito, troviamo la creazione di una legenda, operazione semplicissima essendo sufficiente dichiarare il testo descrittivo che desideriamo associare alla curva *dopo* averla tracciata, con il comando `\addlegendentry`. La posizione della legenda, generata in automatico può essere scelta con l'opzione legend pos.

```
\documentclass{standalone}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[italian]{babel}
\usepackage{pgfplots}

\pgfplotsset{%
every axis plot/.append style={line width=1pt}}

\begin{document}
\begin{tikzpicture}
\begin{semilogxaxis}[
    xlabel={Frequenza media nel periodo,
            $\lambda=1/T_\mathrm{R}$},
    ylabel={$p_{V_\mathrm{R}}$ probabilità
```

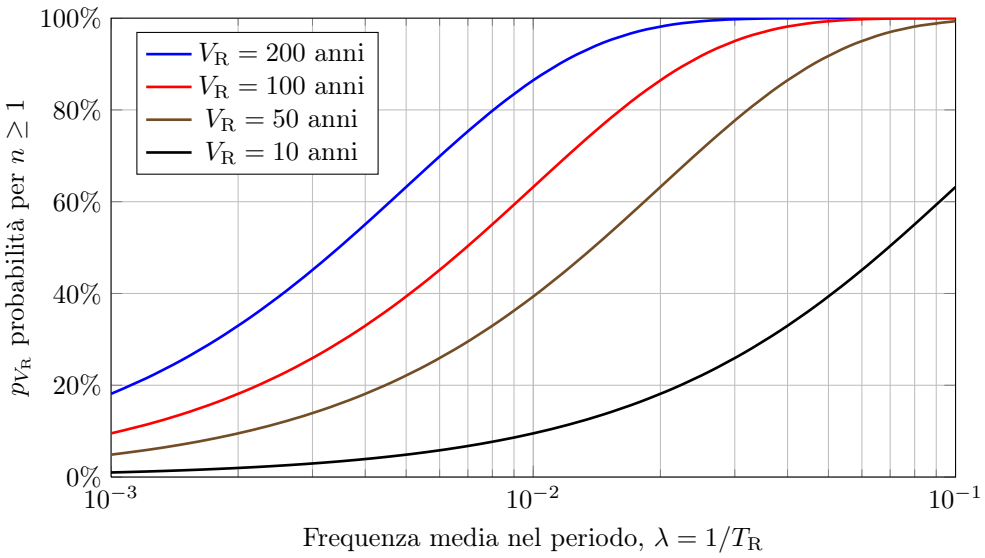


FIGURA 14: Un tipo di grafico semilogaritmico in ascissa

```
          per $n\geq1$},
    ytick={0,0.2,0.4,0.6,0.8,1},
    yticklabels={0\%,20\%,40\%,60\%,80\%,100\%},
    grid=both,
    xmin=0.001, xmax=0.1,
    ymin=0, ymax=1,
    domain=0.001:0.1,
    no markers,
    width=12.75cm,
    height=7.65cm,
    smooth,
    legend pos=north west]

\addplot gnuplot[id=pvriv] {1-exp(-x*200)};
\addlegendentry{$V_\mathrm{R} = 200$ anni};

\addplot gnuplot[id=pvriii] {1-exp(-x*100)};
\addlegendentry{$V_\mathrm{R} = 100$ anni};

\addplot gnuplot[id=pvrii] {1-exp(-x*50)};
\addlegendentry{$V_\mathrm{R} = 50$ anni};

\addplot gnuplot[id=pvri] {1-exp(-x*10)};
\addlegendentry{$V_\mathrm{R} = 10$ anni};
\end{semilogxaxis}
\end{tikzpicture}
\end{document}
```

L'uso di gnuplot[8] necessita di alcune spiegazioni di carattere operativo: il pacchetto PGFPLOTS genera un file contenente il codice nel linguaggio gnuplot che a sua volta genera il file contenente i dati numerici della curva. L'esecuzione di gnuplot sul primo file, richiede la compilazione del sorgente LaTeX con l'opzione `-shell-escape` per concedere il permesso di esecuzione di programmi esterni.

Una volta generati i dati con una prima compilazione, tipicamente con pdflatex, non ci sarà più bisogno dell'esecuzione esterna almeno fino a quando non si modifica l'espressione matematica richiesta[9].

7.6 Piano polare

Con i recenti aggiornamenti del pacchetto PGFPLOTS si è aggiunto il tipo di grafico polare. Le coordinate su questo tipo di piano sono una coppia ordinata di numeri in cui il primo è l'angolo, chiamato anche anomalia, ed il secondo è il raggio. Nella sintassi del comando `\addplot` le grandezze polari vengono ancora riconosciute con i nomi classici cartesiani, `x` ed `y`, rispettivamente per angolo e raggio vettore.

La matematica spesso è sinonimo di bellezza, specie se la possiamo apprezzare visivamente in un grafico. La spirale logaritmica è un ottimo candidato al giudizio estetico ed in natura la si trova nella forma di alcune galassie ed in quella della conchiglia del Nautilus.

Invece di disegnare una spirale logaritmica qualunque, ne disegneremo più di una di tipo a spirale aurea caratterizzata da un particolare valore della costante d'esponente. Il risultato è la figura 15 prodotta dal codice seguente:

```
\documentclass{standalone}
\usepackage{pgfplots}
\usepgfplotslibrary{polar}

\begin{document}
\begin{tikzpicture}
\begin{polaraxis}[
  xticklabels=\empty,
  ytick={0,40,...,400},
  yticklabels=\empty,
  width=12cm,
  samples=600,
  mark=none,
  domain=0:720]

\foreach \phase in {180,200,220}
   \addplot[line width=.65pt,blue!60]
```

8. Ovvio che gnuplot deve essere installato sul sistema.
9. Nell'esempio, abbiamo assegnato un identificatore per ciascuna curva che viene utilizzato nei nomi dei file generati dietro le quinte.

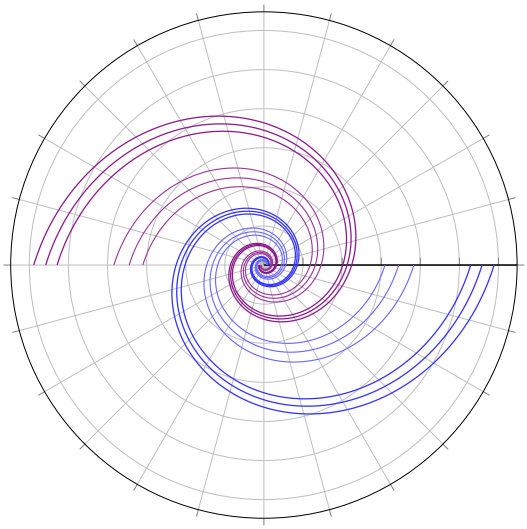


FIGURA 15: Intreccio galattico: una serie di spirali logaritmiche *auree* nel piano polare di PGFPLOTS

```
        {exp((x+\phase)*0.0053548)};

\foreach \phase in {280,290,300}
   \addplot[line width=.75pt,blue!80]
        {exp((x+\phase)*0.0053548)};

\foreach \phase in {180,200,220}
   \addplot[line width=.65pt,violet!80]
        {-exp((x+\phase)*0.0053548)};

\foreach \phase in {280,290,300}
   \addplot[line width=.75pt,violet!90]
        {-exp((x+\phase)*0.0053548)};
\end{polaraxis}
\end{tikzpicture}
\end{document}
```

7.7 Grafici ad istogramma

Alcune tipologie di dati vengono tradizionalmente rappresentati con barre di altezza proporzionale al valori. Sono i grafici ad *istogramma*, supportati da PGFPLOTS in modo molto semplice come variante di visualizzazione del comando `\addplot` con l'usuale proprietà chiave/valore. Se si è interessati ad un istogramma a barre verticali l'opzione citata da fornire è ybar.

In figura 16 è visibile un semplice grafico ad istogramma, generato dal codice del seguente listato, che rappresenta una sequenza di risultati sperimentali che assomiglia ad una distribuzione statistica gaussiana.

```
\documentclass{standalone}
\usepackage{lmodern}
\usepackage{pgfplots}
\usepackage[output-decimal-marker={,}]{siunitx}

\begin{document}
\begin{tikzpicture}
\begin{axis}[
   % etichette del grafico
     xlabel={Numero provino},
     ylabel={Compressione $f_{c}$ (\si{N/mm^2})},
```

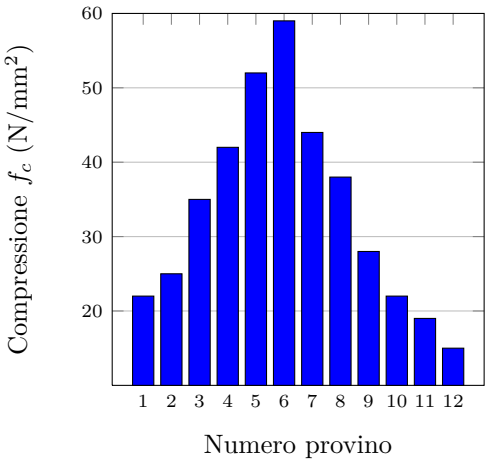


FIGURA 16: Un esempio di un istogramma

```
     tick label style={font=\scriptsize},
     xtick={1,2,...,12},
     ytick={20,30,...,60},
     ymajorgrids=true,
   % dominio 2D
     ymin=10, ymax=60,
   % dimensioni tela
     width=6.5cm, height=6.5cm,
]
\addplot[ybar,bar width=8pt,fill=blue,draw=black]
        coordinates {(1,22)(2,25)(3,35)
                     (4,42)(5,52)(6,59)
                     (7,44)(8,38)(9,28)
                     (10,22)(11,19)(12,15)};
\end{axis}
\end{tikzpicture}
\end{document}
```

Ad ogni risultato di prova corrisponde una coppia di coordinate in cui l'ascissa fittizia è il numero progressivo della barra, trascritto poi in etichetta sull'asse orizzontale e l'ordinata è l'altezza della barra.

7.8 Generazione di grafici ad alto livello

Negli esempi visti finora la generazione numerica dei dati da plottare è stata gestita all'interno dell'ambiente axis sfruttando il motore matematico interno PGFmath oppure programmi esterni come gnuplot (supportato da PGFPLOTS), oppure ancora specificando esplicitamente le coordinate. Esiste tuttavia un diverso *punto di vista* del tutto generale e basato sul fatto che i sorgenti TEX, essendo file in formato testo, possono essere facilmente costruiti da altri programmi.

Si tratta di una modalità per la quale un linguaggio specifico svolge un ruolo di interfaccia ad alto livello verso TEX con cui si interagisce indirettamente in un ambiente operativo anche completamente differente come una finestra grafica che presenta dati modificabili utilizzando il mouse.

7.8.1 Primo passo: curve di Euler-Johnson

Il primo passo applicativo di questa idea è costruire la sequenza di coordinate da plottare con un

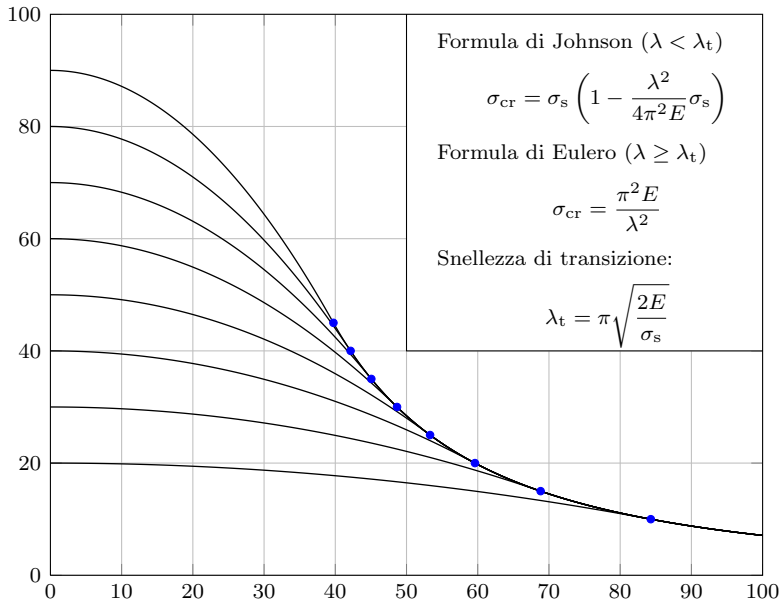


Figura 17: Curve di Euler-Johnson da file dati costruiti in Lua

linguaggio di programmazione. La risoluzione del problema è così affidata ad un componente software completamente indipendente da TEX che genererà un opportuno file tabellare contenente il risultato dell'elaborazione numerica che servirà da input in un usuale sorgente PGFPLOTS.

L'argomento[10] che utilizzeremo per sperimentare la procedura, tratta dalla stabilità dell'equilibrio secondo il modello di Euler-Johnson. Lua svolgerà il ruolo di *linguaggio esterno* per la sue caratteristiche che lo rendono perfetto per questo compito.

Lua è infatti un linguaggio molto semplice da utilizzare perché minimo nella sintassi e perché possiede la tabella, l'unica struttura dati disponibile, ma al contempo è efficiente nell'esecuzione e facile da adoperare non essendo necessario compilare alcun che.

Il nostro programma in Lua dovrà calcolare una funzione matematica per punti e scrivere il risultato in un file di testo stampando riga per riga i valori di ascissa ed ordinata separati da un carattere di tabulazione. Questo formato dei dati è direttamente importabile da PGFPLOTS.

Il listato Lua sotto riportato crea in realtà più file dati relativi a diverse curve ciascuna dipendente da un parametro e definita su due intervalli adiacenti da diverse espressioni matematiche: nel primo ramo useremo il modello di Johnson mentre nel secondo quello classico di Eulero.

Per le note tecniche di Lua, diciamo che i cicli `for` per il calcolo utilizzano una variabile intera per evitare i problemi di approssimazione sul contatore in quanto Lua usa per i numeri interi e per quelli in virgola mobile lo stesso tipo di dato. In particolare, il valore finale del contatore in virgola mobile potrebbe essere leggermente superiore al valore intero di confronto causando un comportamento inaspettato del ciclo.

Inoltre, si è preferito memorizzare i dati in tabelle anziché scrivere i dati immediatamente sul file una riga alla volta, per poi trasferirne l'intero contenuto un unica operazione di scrittura. Esiste infatti la funzione `table.concat()` della libreria standard di Lua `table`, che restituisce la stringa risultato della concatenazione dei singoli elementi della tabella passata come parametro.

La scelta più che per motivi di efficienza (la quantità di dati dell'esempio non avrebbe dato problemi di efficienza in ogni caso) è stata fatta per motivi didattici, per mostrare come Lua sia in grado di gestire assolutamente senza problemi stringhe di grandi dimensioni, e si è così evidenziato come sia preferibile scrivere su file una volta sola per la migliore gestione delle operazioni di input/output su disco.

Altro vantaggio della concatenazione degli elementi stringa di una tabella è che si evita il problema che coinvolge lo spostamento di grandi blocchi di memoria dovuto alla concatenazione diretta delle stringhe.

Il calo di prestazioni nel nostro caso non si sarebbe comunque avvertito se avessimo implementato l'idea di conservare i dati in una stringa invece che in una tabella. Ad ogni passo del ciclo avremo potuto infatti concatenare alla stringa che contiene i dati fino a quel momento calcolati, quella della nuova riga numerica incorrendo nel problema di memoria.

In Lua, come in molti altri linguaggio per esempio in Java, le stringhe sono tipi *immutabili* cioè non possono essere modificate. La concatenazio-

10. Questo esempio è un contributo graditissimo di Orlando Iovino. Grazie ansys...

ne di due stringhe provoca quindi la creazione di una nuova stringa e non un aggiornamento di un oggetto testuale. Ad ogni concatenazione tutto il blocco di memoria della stringa dei dati verrebbe ricopiato, movimentando porzioni di dati sempre più grandi.

Finalmente, ecco il listato in Lua che soddisfa le precauzioni di efficienza menzionate e quindi utile per essere un esempio di codice riusabile in altri casi simili di elaborazione numerica:

```
--[[
   Curve di Eulero-Johnson per vari valori
   del carico di snervamento del materiale
--]]

-- funzione ausiliaria scrittura dati
local function formatData(x,y)
   return string.format("%.2f\t%.3f",x,y)
end

-- funzione ausiliaria creazione file
local function makefile(fn,t)
   local outputfile = assert(io.open(fn,"w"))
   outputfile:write(table.concat(t,"\n"))
   outputfile:close()
end

-- Modulo di Young (daN/mm^2)
local EY = 7200

-- intervallo di calcolo e numero di suddivisioni
local Lmin, Lmax, n = 0, 100, 1000
local step = (Lmax-Lmin)/n

-- tabella dati di transizione
local points = {}

for ss=20,90,10 do -- valore iniziale, finale, e passo
   -- tabella temporanea dati
   local t = {}

   -- lunghezza di transizione
   local Lt = math.pi*math.sqrt(2*EY/ss)

   -- passi fino a non superare Lt
   local Ltlim = math.floor((Lt-Lmin)/step)

   -- Rottura alla Johnson
   for i = 0, Ltlim do
      local L = Lmin + i*step
      local J =ss*(1-(L^2)/(4*math.pi^2*EY)*ss)
      t[#t+1] = formatData(L,J)
   end

   points[#points+1] =
            formatData(Lt,(math.pi^2*EY)/(Lt^2))

   -- Rottura alla Eulero
   for i = Ltlim+1, n do
      local L = Lmin + i*step
      local E =(math.pi^2*EY)/(L^2)
      t[#t+1] = formatData(L,E)
   end

   -- scrittura della tabella su file esterno
   makefile("dati"..tostring(ss)..".txt", t)
end

makefile("points.txt", points)
```

Passando al sorgente LaTeX che genera il grafico di figura 17, notiamo che il tracciamento delle curve è prodotto da un codice molto elegante che usa il costrutto `\foreach` del pacchetto PGF per leggere ad ogni ciclo il file dati opportuno con il comando `\addplot` con la direttiva `file`. L'unica attenzione è quella di coordinare i nomi dei file, ma l'effetto è notevole: due righe di codice sono sufficienti per disegnare tutte le curve.

Altra caratteristica del grafico è il disegno all'interno della tela, di un breve testo esplicativo della formulazione matematica rappresentata dalle curve stesse, risolto con un istruzione PGF che disegna un rettangolo bianco ed un oggetto nodo a cui si assegna una scatola di testo. La sequenza di oggetti elaborata dal comando `\draw` è un path dunque il nodo viene posizionato nel secondo *angolo* del rettangolo imponendone come dovrà essere definito in termini di coordinate.

```
\documentclass{standalone}
\usepackage{amsmath}
\usepackage{pgfplots}

\def\explainmathbox{%
\rule{8pt}{0pt}\parbox{4.5cm}{
\footnotesize\smallskip
Formula di Johnson ($\lambda<\lambda_\mathrm{t}$)
\[\sigma_\mathrm{cr}=
  \sigma_\mathrm{s}\left(1-\frac{\lambda^2}
{4\pi^2E}\sigma_\mathrm{s}\right)
\]
Formula di Eulero ($\lambda\geq\lambda_\mathrm{t}$)
\[\sigma_\mathrm{cr}=
  \dfrac{\pi^2 E}{\lambda^2}
\]
Snellezza di transizione:
\[\lambda_\mathrm{t}=
  \pi\sqrt{\dfrac{2E}{\sigma_\mathrm{s}}}
\]
}}

\begin{document}
\begin{tikzpicture}
\begin{axis}[tick label style={font=\footnotesize},
             width=110mm, height=90mm,
             xmin=0, xmax=100,
             ymin=0, ymax=100,
             grid=major
             ]

\foreach \ss in {20,30,...,90}
   \addplot[line width=0.5pt] file {dati\ss.txt};

\addplot[mark=*,
         mark size=1.4pt,
         only marks,
         draw=blue,
         fill=blue] file {points.txt};

\draw[fill=white]
     (axis cs:100,40) rectangle (axis cs:50,100)
     node[anchor=north west]
     {\explainmathbox};
\end{axis}
\end{tikzpicture}
\end{document}
```

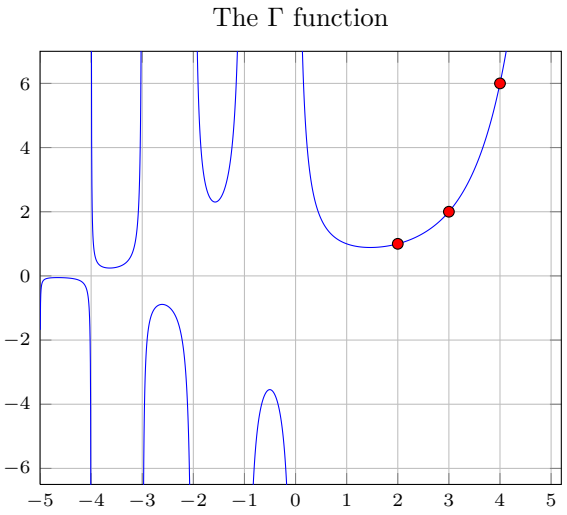


FIGURA 18: La funzione Γ di Eulero

7.8.2 *Secondo passo: la Gamma di Eulero*

Il secondo passo è dare al programma esterno anche l'onere della creazione del sorgente LaTeX. L'utente darà quindi le opportune istruzioni per generare i file dati, sempre in formato testo, e quelle per scrivere su disco il sorgente che li utilizzerà, procedendo eventualmente anche alla compilazione. Un qualcosa di molto simile è già stato esemplificato dallo stesso Autore di PGFPLOTS in un piccolo script in Python disponibile in CTAN in una delle sotto cartelle dell'albero dedicato al pacchetto, denominata script.

Poiché la funzione che ci siamo prefissati di disegnare per questo secondo obiettivo è la funzione Γ di Eulero, nota come l'estensione al campo dei numeri complessi del fattoriale di un numero naturale, prepariamoci ad affrontare un percorso non facile considerato il comportamento asintotico della funzione trovata nel 1729 dall'allora ventiduenne Eulero su sollecitazione di Christian Goldbach.

Occorre avvalerci di librerie numeriche particolari come per esempio il modulo Python *special* del pacchetto scientifico scipy (VARI, 2011). Faremo quindi un esempio anche con questo linguaggio interpretato molto simile a Lua, senza tuttavia entrare nei dettagli della sintassi che si fonda sull'identazione del codice come mezzo per individuare i blocchi di istruzioni.

Per facilitare la gestione dei dati da parte di PGFPLOTS formeremo dapprima il file di dati contenente le coppie di valori ⟨*ascissa*⟩ ⟨*ordinata*⟩ della funzione Γ nell'intervallo previsto dell'asse reale e poi il file sorgente completo delle opzioni dell'ambiente axis e del tracciamento dei punti corrispondenti ai fattoriali dei primi tre numeri naturali. Il risultato del codice riportato di seguito è la figura 18.

```
#!/usr/bin/python

# import delle librerie necessarie
import os
import numpy as np
from scipy.special import gamma as Gamma

def add(v):
        """Little more simply 'append' function"""
        content.append(v+'\n')

# definizione dei limiti del grafico
# per il facile 'aggiustamento'
x0, x1 = -5.0, 5.2
y0, y1 = -6.5, 7.0

# dominio lineare sull'asse x
x = np.arange(x0, x1, 0.005)

# creazione dei vettori di ordinata
y = Gamma(x)

# 'gestione' delle discontinuita
y[y>100*y1] = 'inf'
y[y<100*y0] = 'inf'

# costruzione dati di plottaggio
data = []
data.extend(''.join(['{0} {1}\n'.format(u, v) \
                    for u, v in zip(x,y)]))

# scrittura file dati
f = open('data.txt', 'w')
f.writelines(data)
f.close()

# creazione sorgente TeX
content = []
add('% build by a Python script\n')
add('\\documentclass{standalone}')
add('\\usepackage{lmodern}')
add('\\usepackage{pgfplots}')
add('\\begin{document}')
add('\\begin{tikzpicture}')
add('\\begin{axis}[')

# opzioni grafico
add('title={The $\\Gamma$ function},')
add('tick label style={font=\scriptsize},')
add('xmin='+str(x0)+',')
add('xmax='+str(x1)+',')
add('ymin='+str(y0)+',')
add('ymax='+str(y1)+',')
add('xtick={-5,...,5},')
add('grid=major,')
add('unbounded coords=jump')
add(']')

# plottaggio
add('\\addplot+[no markers] file {data.txt};')
add('\\addplot[mark=*,only marks,')
add('          fill=red] coordinates {')
add('(2,1)')
add('(3,2)')
add('(4,6)')
add('};')

# chiusura sorgente
add('\\end{axis}')
add('\\end{tikzpicture}')
add('\\end{document}')

# scrittura file sorgente
filename = 'graficoGamma'
s = open(filename+'.tex', 'w')
```

```
s.writelines(content)
s.close()

# compilazione del grafico
os.system('pdflatex ' + filename)
os.remove(filename +'.aux')
os.remove(filename +'.log')
```

7.8.3 *Passo tre: verso un modulo* ad hoc

Nel primo passo illustrato in questa sezione il grafico è il risultato di due componenti distinti, un programma che si occupa di costruire i dati ed un sorgente LaTeX che si occupa della loro rappresentazione.

Con il secondo passo queste due componenti sono unite in un unico programma che vede TeX come una sorta di libreria.

Il terzo passo aggiunge ai precedenti un ulteriore salto concettuale: la costruzione di un vero e proprio *linguaggio* che vive all'interno di un programma in cui scompare totalmente ogni riferimento a TeX od al pacchetto PGFPLOTS.

L'idea non è certo nuova essendo già descritta dal potente paradigma modello/vista in cui i dati sono indipendenti dal modo in cui possono essere rappresentati, esattamente allo stesso modo per cui un sorgente LaTeX è il *modello* ed il risultato della compilazione è la *vista*.

Ci si è allontanati dal linguaggio di PGFPLOTS in direzione di uno ad un più alto livello di astrazione che prevede concetti inerenti sia ai dati che ai relativi grafici.

La messa a punto di un linguaggio di questo tipo potrebbe trovare il supporto ideale in Python per via della disponibilità di efficienti e complete librerie di calcolo numerico con lo sviluppo di un nuovo modulo tra i molti già disponibili. Per esempio in matplotlib[11], che prevede la costruzione di grafici per mezzo di *oggetti* Python e la successiva loro traduzione in file di uscita per mezzo di componenti chiamati *backend*, ciascuno dedicato ad un formato specifico (per esempio Postscript, PDF, SVG o formati raster come PNG o JPEG).

8 Tecnologie per collaborare

La rete Internet permette ormai da anni di scambiarsi informazioni digitali in forma pubblica o privata indipendentemente dal luogo dove si trovano gli interlocutori. Anche questo articolo è stato scritto utilizzando le tecnologie della rete, concretizzandone le possibilità di collaborazione a distanza in tempo reale.

Più che la descrizione delle tecnologie utilizzate per l'articolo, è importante far notare che molto, forse tutto del futuro della documentazione e della comunicazione delle idee del mondo TeX avverrà con questi strumenti informatici.

11. Si veda l'indirizzo `http://matplotlib.sourceforge.net/`.

Essenziale diventa quindi incoraggiare la collaborazione a distanza via internet e raccogliere le esperienze dei singoli progetti basati sul web, preferibilmente in articoli od altra documentazione specifica, per poi selezionare le tecnologie della rete più adatte alle attività in condivisione sugli argomenti della tipografia digitale.

9 Conclusioni

Come chiaramente illustrano gli esempi il pacchetto PGFPLOTS consente la produzione di grafici di elevata qualità sia tecnica che tipografica. Ciò è possibile grazie a un linguaggio basato su una struttura semplice, con molte possibilità di configurazione degli elementi grafici e sulla possibilità di creazione di nuovi stili.

Non solo, i brillanti risultati che si possono conseguire con l'elegante e potente struttura sintattica di PGFPLOTS, dimostrano implicitamente anche la qualità del pacchetto PGF su cui si basano tutte le funzionalità grafiche.

Per superare le limitazioni intrinseche di TeX nel calcolo in virgola mobile, il pacchetto PGFPLOTS prevede la possibilità di elaborare internamente le funzioni matematiche, spesso con precisione sufficiente, oppure di effettuare i calcoli per il tracciamento dei grafici con programmi esterni, con il supporto a gnuplot.

Dal punto di vista operativo e numerico, il pacchetto consente di accedere ad insiemi di dati numerici in forma tabellare generati con altri applicativi dedicati a risolvere ed implementare i problemi di calcolo. In questa modalità, il pacchetto PGFPLOTS assume il compito di *visualizzatore* dei dati. In questo senso il pacchetto ha le potenzialità per essere impiegato come libreria inclusa in altri componenti software.

10 Ringraziamenti

Ringraziamo l'anonimo revisore della redazione della rivista che ha svolto un preziosissimo lavoro di verifica e miglioramento dei contenuti dell'articolo.

Per il suo contributo, ringraziamo anche Orlando Iovino che ci ha fornito l'esempio sulla costruzione del grafico di Euler-Johnson, l'idea quindi di produrre i dati con un programma in Lua.

Un doveroso Grazie va anche alle famiglie degli Autori che hanno sopportato il ticchettio sulla tastiera al posto di buone conversazioni di casa ed al lettore senza il quale nulla si sarebbe scritto.

Riferimenti bibliografici

Feuersänger, C. (2010). «Manual for package PGFPLOTS». Disponibile su `www.ctan.org`.

Giacomelli, R. (2008). «Una tabella che fa calcoli». *ArsTEXnica*, (6), pp. 28–36. URL `http://www.guit.sssup.it/arstexnica.php`.

Ierusalimschy, R. (2006). *Programming in Lua*. Lua.org, 2ª edizione.

Tantau, T. (2010). «pgf manual version 2.10». Disponibile su `www.ctan.org`.

Vari (2011). «Libreria scientifica in linguaggio python». Sito web `http://www.scipy.org/`.

▷ Agostino De Marco
Università degli Studi di Napoli "Federico II"
Dipartimento di Ingegneria Aerospaziale
`agostino dot demarco at unina dot it`

▷ Roberto Giacomelli
Carrara (MS)
`giaconet dot mailbox at gmail dot com`