Numero 12
Ottobre
2011

ArsTEXnica

Rivista italiana di TEX e LATEX

Atti del GUIT *meeting 2011*

GUIT

http://www.guit.sssup.it/arstexnica

GUIT – Gruppo Utilizzatori Italiani di TEX

ArsTEXnica è la pubblicazione ufficiale del GUIT

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

Le uscite avranno, almeno inizialmente, cadenza semestrale e verranno pubblicate nei mesi di Aprile e Ottobre. In particolare, la seconda uscita dell'anno conterrà gli Atti del Convegno Annuale del GUIT, che si tiene in quel periodo.

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

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

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

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

`arstexnica@sssup.it`.

Nota sul Copyright

Associarsi a GUIT

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

Indirizzi

Gruppo Utilizzatori Italiani di TEX:
c/o Ufficio Statistica
Scuola Superiore Sant'Anna
Piazza Martiri della Libertà 33
56127 Pisa, Italia.
`http://www.guit.sssup.it`
`guit@sssup.it`

Redazione ArsTEXnica:
`http://www.guit.sssup.it/arstexnica/`
`arstexnica@sssup.it`

Codice ISSN 1828-2369

Stampata in Italia
Pisa: 15 Ottobre 2011

Programma del Convegno

Sessione Mattutina

09:00 – Registrazione

09:30 – XeLaTeX and the PDF archivable format
Claudio Beccari

10:00 – LaTeX nella pubblica amministrazione. La web application FACILE per la produzione automatica di comunicazioni interne del Comune di Napoli
Agostino De Marco, Università degli Studi di Napoli "Federico II"
Paolo Eugenio Cresci, Comune di Napoli

10:30 – Introduzione agli strumenti per grecisti classici
Gianluca Pignalberi, Salvatore Schirone
Jerónimo Leal, Pontificia Università della Santa Croce

11:00 – Coffee break

11:30 – Creare grafici con pgfplots
Agostino De Marco, Università degli Studi di Napoli "Federico II"
Roberto Giacomelli

12:00 – Sezione Lavori in corso: Sulla realizzazione di un font Dante di Giovanni Mardersteig
Claudio Vincoletto
Massimiliano Dominici

12:30 – Sezione lavori in corso: collana TeXnologie
Gianluca Pignalberi, Massimiliano Dominici
Silvia Maschio, CompoMat S.R.L.

13:00 – Pausa Pranzo

Sessione Pomeridiana

14:30 – Passare da LaTeX a XSL-FO
Jean-Michel Hufflen, Université de Franche-Comté

15:00 – Extending ConTEXt MkIV with PARI/GP
Luigi Scarso, Logo S.R.L.

15:30 – Discussione sulle prospettive del gruppo

17:00 – Chiusura dei lavori

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 12, Ottobre 2011

Editoriale
Gianluca Pignalberi 5
XeLaTeX and the PDF archivable format
Claudio Beccari 6
Creare grafici con pgfplots
Agostino De Marco, Roberto Giacomelli 12
LaTeX nella pubblica amministrazione.
La web application FACILE per la produzione automatica di comunicazioni interne del Comune di Napoli
Agostino De Marco, Paolo Eugenio Cresci 39
Presentazioni da ArsTEXnica numero 11 57
An Introductive Presentation of XSL-FO
Jean-Michel Hufflen 58

Gruppo Utilizzatori Italiani di TEX

Editoriale

Gianluca Pignalberi

A pochi giorni dalla chiusura tardiva di ArsTEXnica numero 11 mi trovo a chiudere di corsa il numero 12, dedicato al GUIT *meeting* 2011. Questo numero, e questa edizione del meeting, sono in apparenza molto povere: solo tre articoli tre da pubblicare e presentare.

L'apparente povertà è bilanciata dalla qualità dei lavori. L'ineffabile Claudio Beccari ha scritto un articolo dedicato a XeLaTeX. Ci spiega con la solita precisione che lo contraddistingue il perché questo compilatore non è adatto a produrre documenti conformi allo standard PDF/A-1, e come si può fare in modo che li produca. Non occorre qui ribadire l'importanza di questo particolare formato, avendone parlato in un precedente numero di ArsTEXnica.

Agostino De Marco e Roberto Giacomelli hanno scritto un lunghissimo, ma tutt'altro che noioso o ridondante, articolo in cui spiegano come preparare dei grafici dall'aspetto più che professionale con pgfplots. Coprono molti tipi di grafici, fornendo una mole impressionante di esempi e di codice. Nel corso dell'articolo forniscono anche spiegazioni quantitative e tecniche sul perché usare o non usare un certo strumento o una certa tecnica. Un articolo da leggere anche al di là dell'interesse specifico.

Sempre Agostino De Marco, ma stavolta con Paolo Eugenio Cresci, ci stupisce con un articolo in cui dimostra come LaTeX trova impiego... al Comune. Il Comune di Napoli, nell'ambito di un progetto molto articolato e di sicuro interesse per tutti gli enti pubblici italiani, porta avanti un progetto di *corporate identity* in cui LaTeX serve a comporre delle lettere in maniera standardizzata e tipograficamente ineccepibile. Certo, molti dei dipendenti non sapranno che LaTeX è il loro motore di impaginazione, ma sono convinto che molti utenti della TV satellitare ignoreranno di avere un sintonizzatore pilotato da GNU/Linux e da altri programmi aperti.

Questo è quanto per gli interventi al meeting i cui articoli sono pubblicati nel numero che state leggendo. Siccome un meeting con tre lavori sarebbe improponibile, abbiamo ripescato un paio di articoli del numero 11 di ArsTEXnica i cui autori hanno dato disponibilità a fare una presentazione. Trovate i sommari, originali o estesi, nelle pagine finali.

Infine, non c'è bisogno di ribadirlo, il GUIT è una realtà mai immobile, né come associazione, né come attività dei singoli membri. Questa edizione del meeting (l'ottava, per la precisione) vede l'introduzione di una sezione Lavori in corso, in cui gruppi di lavoro più o meno formali presentano dei progetti in fase embrionale o di sviluppo, ma non ancora maturi per un articolo. Che il progetto sia portato avanti a nome e per conto del GUIT o no poco importa; per tutti il denominatore comune è TeX. Non parlerò ora di alcuno di questi progetti, riservandomi di farne un resoconto sul prossimo numero di ArsTEXnica, che spero potrà ospitare anche degli articoli proprio sui progetti presentati.

Troverete una novità nella Call for Papers: la consegna degli articoli viene anticipata di un mese. Questo perché tutti i redattori, al momento, sono lavorativamente impegnati (c'è da dire "meno male", di questi tempi), ed è bene allungare un po' i tempi di lavorazione. Spero con questo stratagemma di chiudere per tempo il numero 13 della nostra rivista.

Non mi pare il caso di procrastinare; vi lascio alle letture importanti. Vi auguro buon meeting (per chi c'è) e vi aspetto sul prossimo numero.

▷ Gianluca Pignalberi
`g dot pignalberi at alice dot it`

XeLaTeX and the PDF archivable format

Claudio Beccari

Sommario

Up to now XeLaTeX produces a final PDF output file but it gets this output by means of the transforamtion of a XDV (extended DVI) intermediate file. This excludes most of the possibilities offered by pdfLaTeX that, at least since version 1.40.9, and with the help of an extension file pdfx.sty, can directly producce a PDF/A-1b compliant output. This paper explains how to get through this bottle neck by resorting to the ubiquitous ghostscript program.

Sommario

Fino ad oggi XeLaTeX produce un file finale in formato PDF ma passando attraverso la trasformazione di un file intermedio in formato XDV (extended DVI). Questa trasformazione esclude che ci si possa servire delle funzionalità del programma pdfLaTeX che, almeno dalla versione 1.40.9 in poi, con l'aiuto di un file di estensione pdfx.sty, riesce a produrre direttamente un file di uscita conforme allo standard PDF/A-1b. Questo articolo spiega come superare questo intoppo ricorrendo al programma ghostscript presente su ogni piattaforma.

1 Introduction

Several papers have been already published on several TeX related magazines, ArsTeXnica included, about producing PDF/A compliant archivable files. Almost all these papers focused the reader's attention on the various *caveats* that is necessary to observe in order to avoid the many pitfalls of this format. Some papers focused the attention also on the fact that the color profiles are not so clearly defined so that sometimes a non compliant file is obtained just because an unsuitable color profile file has been employed. Some papers focused the attention also on the fact that sometimes the Preflight program, the most authoritative one included in the Adobe Acrobat Pro suite, fails to recognize the file compliance with the ISO standard labelled PDF/A-1a or PDF/A-1b. But to my best knowledge, no paper deals with producing a PDF/A compliant file from a source intended to be composed with XeLaTeX.

Let us recall some pieces of information. The PDF/A ISO standard has been devised in the year 2005 and regulated with the ISO-19005-1:2005 regulation. This regulation standardizes two subformats, labelled 1a and 1b; the latter is less stringent than the former; it requires that the level of the PDF language used in the PDF file is level 4; it requires the fonts to be vector ones and that they are subset-embedded in the document file; it requires that the color profiles are clearly defined, and it requires the presence of a certain number of *metadata* to be included into the file in a non encrypted form so that library searches can be performed. The purpose is to have files that fulfill specific limitations, but that will be readable from now on for an unlimited length of time, in spite of the fact that in, say, fifty years the fonts and the programs available today may not exist any more. The former sub format, the 1a one, is more stringent in the sense that the fonts must be completely embedded and also the document structure must be included in the file.

As far as I am aware of, archivable files in the 1b sub format are generally sufficient for the purpose of the long term reproduction on screen of the archived documents, exactly as the authors intended them to be. Therefore I will concentrate on this "simpler" format, also because I did not find any means for producing the 1a format, except the Preflight program of Adobe Acrobat Pro, which I have no direct access to.

Finally it must be noticed that a new standard has been issued in 2011, ISO 19005-2:2011, that extends a little bit the previous PDF/A standard; with this new standard the PDF language level 6 may be used and the JPEG2000 images may be included in the archivable documents. Even if these are important enhancements in certain areas, I will not deal with them and stick with the previous standard, for no other reason than the ghostscript program, needed for the task, is not yet capable of satisfying the new standard. I am sure that in a short time even ghostscript is updated and that its documentation shows the small modifications that need to be introduced into the necessary scripts.

2 The PDF/A requirements

2.1 The *metadata*

Any PDF/A compliant file must contain some *metadata* that describe some features of the document, from the color profile and the document title and author, to the keywords that ease the library search of the archived document. Some *metadata* are compulsory, some are optional.

For what concerns the compulsory *metadata* I show below how to prepare a suitable auxiliary file

that contains all the necessary information in the PostScript language; ghostscript will translate such information into the XML language and will insert this XML code into the output file.

In a certain sense this is the simplest part of the whole procedure; the problem consists in knowing what information must be supplied and in which form.

2.2 The color profile

One of the most misterious pieces of information is the one that describes the color profile; Luigi Scarso already wrote a paper (Scarso, 2010) where he discusses this problem in connection with the typesetting program ConTEXt-mkiv. But the problem is not the particular typesetting program, rather it is the very concept of *color profile*. I won't go any deeper into this question, but I redirect the reader on the paper Scarso (2010), where the question is thoroughly discussed. I just remark that I have obtained satisfactory results by downloading the color profile file ECI-RGB.V1.0.icc, freely downloadable from the download area of the www.eci.org Internet site. This file may be saved on one's disk somewhere, where the ghostscript program can find it; but I suggest to save this file in a system wide folder and to specify the full address when dealing with this file.

The mentioned color profile is a generic one, and yields satisfactory results in most circumstances; of course it only deals with the RGB (Red, Green, Blue) additive color model (the one that is used by the TV and computer screens) and the pictures inserted into the document file should be consistent with this color model; therefore no picture in the CMYK (Cyan, Magenta, Yellow, blacK) subtractive color model should be used in a PDF/A compliant file when the color profile refers to the RGB color model.

2.3 Hyperlinks

PDF/A compliant files may use internal hyperlinks in order to ease the document navigation; internal links means that the link targets are internal to the document; therefore the reader may use the bookmark column of the PDF reader so as to jump from one point to another of the document, and this possibility is particularly useful in a reference document.

External links are prohibited; external links refer to other documents on the same computer, or to targets in the Internet. It is evident that a long term archiving process cannot have links between objects that do exist today, but most likely will not exist any more in fifty years. Therefore when archiving is a requirement, any document should be self contained; if one needs to refer to another document, either uses a complete reference in the document bibliography, or includes the referred document in his/her own one. The choice depends on the author, but s/he must keep in mind the requirement that her/his archived document *must* be self contained.

This is not a trivial constraint; after all it's our daily experience, if we are used to surf the net, that many Internet addresses are not valid today, not to mention fifty years from now!

In order to be sure to have the hyperlinks behave as they should with PDF/A compliant documents, it is sufficient to specify the pdfa option either to the class itself, or to the hyperref package directly, or, even better, by means of the \hypersetup command which the package performance is customized with.

2.4 Fonts

The default fonts of any TEX distribution are pretty good for most purposes, but they fail in some other instances. I already wrote a short paper on this subject (Beccari, 2010) in order to find a patch to "heal" the cmsy* fonts that use some zero-width glyphs. This happened with the third math family fonts, the one that contain only symbols, because some symbols, such as $\mapsto$ and all the negated relation operators $\neq$, $\notin$, etc., resorted to the superposition of a zero-width glyph over, or besides, another symbol. Any zero-width glyph is not PDF/A compliant.

With XeLaTeX the font choice is much wider, although for what concerns math typesetting up to now there are few OpenType UNICODE encoded fonts suitable for the task; but as UNICODE encoded fonts, they are (or should be) safe under the point of view of compliance with the PDF/A requirements. I confess that I did not test for this conformity the recently available OpenType Latin Modern Math fonts, but I tested the XITS Math fonts (those that have the widest choice of math glyphs) and I did not find any glitch.

On the opposite I found out something that either I neglected in the past or that more modern checking programs can spot; it's the "normal" Type 1 Computer Modern OT1 encoded fonts that have some problems with the use of accents.

Evidently if you use XeLaTeX you have no problem in avoiding the traditional Computer Modern fonts; if you like their shape, you'd rather use the OpenType UNICODE encoded CM-Unicode fonts (that contain the full accented set of Latin letters, the Cyrillic ones and the full accented set of Greek letters, suitable for the Greek polytonic spelling), and there would not be any problem. But the problem might show up when you include in your document either PDF files or pages extracted from other documents, when the latter ones were typeset by means of the default CM fonts.

I may have overlooked this problem in the past, because I never use OT1 encoded fonts for typesetting my documents; moreover nobody should actually use the OT1 encoded fonts if the docu-

ment they typeset contains even a single accented word. In facts the poor performance of the OT1 encoded fonts depends on the fact that accents are superimposed on any letter that has to be accented by an overlay mechanism that apparently is acceptable when a document is printed, but in effects is a poor patch that does not follow the best practice used by the OpenType fonts and the T1 encoded "normal" TEX fonts: OpenType fonts use accented letters and T1 encoded fonts translate the accenting process into a glyph substitution so that no overlay process is involved.

Therefore it is most important to check the type and quality of the fonts embedded into a PDF file to be included, be it a technical diagram or a stretch of text. For this purpose I find extremely useful the free program Adobe Reader X (actually any recent version will do the task), where by pressing the Ctrl +D (or the cmd + D key on Mac computers) a dialog pane is opened where the user can select the "Fonts" tag and get the full list of the fonts contained in the document. In particular the user can control that no Type 3 (bitmapped) fonts are used and, if any CM font is being used, the user may examine the document in order to find out if any accented letters have been used. In the latter case the user should process the document to be included by following the procedure shown in the next section.

2.5 Included documents and files

If the documents are in PDF format, the Adobe Reader procedure indicated above may be used also for checking other document characteristics. This or other programs should be used also for controlling the color model of the documents or pictures to be included.

The user must distinguish between bitmap and vector images. The former may contain anything, from photographs to text and line art; they must be controlled only to determine the color model; should it be a "wrong" model almost any image processing program can be used to open the file and save it again with a different color model; remember the only color model usable for PDF/A compliant images is the RGB one; it is admissible also the "gray" model, since the RGB one can render the gray nuances very well. But the bitmapped format may be acceptable for photographs with a pixel density of at least 300 dpi, but in general it is not valid for line art and textual files. The compression method used by the JPEG (`.jpg`) is not a lossless one, therefore the reproduced image may exhibit unwanted artifacts, often very visible if the image contains periodic patterns. The Portable Network Graphics format (`.png`) uses a different compression method and in general it is more suited for line art. Vector graphics, in any case, is more suited for line art even if sometimes it's not possible to have available every line art picture in such a format.

But stretches of text and vector graphics loose their quality in a terrible way if they are converted to bitmapped form; therefore the user should pay much attention to what s/he does with files to be included that do not comply with the PDF/A standard.

For what concerns `.eps` vector files it's easy to convert them to `.pdf` ones, but it's necessary to pay attention that every possible font glyph is embedded into the transformed file.

One way to correct the font problem of a PDF file is to open it with the free program inkscape and save it back to PDF format. In this process the vector fonts are converted into vector drawings that reproduce the same glyphs, but do not exploit nor exhibit any font property. In this way, for example, it is possible to draw the glyphs of the fonts that were not embedded into the file, or it is possible to draw the poor accent overlay of the CM fonts. I confess I never tried this procedure, but it is Hàn Thế Thành himself (Hàn Thế Thành, 2008) who suggests this procedure.

3 The *metadata* auxiliary file

We are almost ready to produce a PDF/A compliant file from a file typeset with XeLaTeX. I assume that XeLaTeX has been run the sufficient number of times so as to be sure that the final PDF document does not require any further adjustment. Of course we can repeat the transformation, but it's better to wait until the document is substantially stable. We are going to use ghostscript; it must be version 8.60 or higher; the more recent the better. At writing time I am using version 9.02.

As already said, we need an auxiliary file in order to introduce the *metadata* and to set up some special PostScript commands necessary for this task. The ghostscript distribution contains a file named `PDFA_def.ps` and is contained in the ghostscript sub tree `.../ghostscript/⟨version⟩/lib/` (of course change the slash character into a backslash one on Windows platforms); where this sub tree is grafted onto your general system tree depend on your platform and your operating system, but you certainly can execute a search command and find out where all this resides.

Copy the above mentioned file `PDFA_def.ps` to the folder where you saved your document master file and suppose this master file is named `myXeLaTeXfile.tex`; copy the above `.ps` file changing its name to `myXeLaTeXfile-def.ps` (notice I changed the underscore into a normal "hyphen sign" or "short dash" sign).

Now open this file with an ASCII editor; since it is a `.ps` file you might right-click the file name, but you have to choose the editor name yourself, because you cannot use the default application for `.ps` files. On a Window platform you might use

Wordpad; on a Linux one you might use vim, emacs, gedit; on a Mac one you might use Textedit.app, TextWrangler.app, but avoid using TeXShop.app (although you may use TeXworks.app) since this programs, besides the TEX system files, may open also the PDF files, and therefore it is capable of "distilling" a `.dvi` or `.ps` file into PDF format, which generally is a very useful feature, but not in this case.

The newly created `myXeLaTeXfile-def.ps` contains some lines commented with a `% Customize` comment; you should edit these lines the way that is best suited to introduce the necessary *metadata* information. In order to insert the *metadata* suitable for this article I would use the file on page 10.

As you can see there are three sets of data that can be customized:

1. The `/ICCProfile`; I changed the default one `ISO Coated sb.icc` into the above mentioned `ECI-RGB.V1.0.icc` and I expressed the full path from the root of my disk to the file; probably I might have used the home folder indicating it with `~`, but the full path is more easily changeable on those Windows platforms that do not have the notion of "home".

2. The document data `/Title`, `/Author` and `/Subject`; other similar *metadata* may be added in a similar way; but remember: these *metadata* have nothing to do with those you can specify in the input files to be processed with pdfLaTeX by means of specific `\pdf...` commands; some of those commands may have a meaning also for XeLaTeX, but their contents does not migrate to the proper PDF/A file section where *metadata* should be.

3. The `/OutputConditionIdentifier`; I did not modify it at all.

Now the fact that this auxiliary file has the same name as the main file of the document comes handy because it is sure that every document has its own auxiliary file and there is no possibility of fiddling with a myriad of `PDFA_def.ps` files, all with the same name, even if they are in different folders, and with ghostscript looking for it starting with its own sub tree; you may get very frustrated trying to figure out why ghostscript does not fetch your newly created `.ps` file.

4 The script

ghostscript requires a long series of specifications for doing its job; the bash scripts (for UNIX-like machines) or batch files for Windows platforms come handy for specifying the whole command string without errors and without the risk of forgetting some terms.

The script on page 10 is a bash script where the only attention to be made is to eliminate the end-of-lines in the last long command that must consist in just one line. In order to change the script into a batch file, it's sufficient to use the Windows name for the ghostscript executable (gswin32c in place of gs) and to use `%1`, `%2`, etc. in place of `$1`, `$2`. Comment signs `#` must be changed into the string `REM`. The first three `file` assignments are not really necessary: one might avoid them and use only one symbolic parameter, but they might be used for testing the presence of the files that are required and to issue the necessary messages in case they were missing. The bash file might be saved with the name pdf2pdfa (or pdf2pdfa.bat with Windows) without forgetting to assign it the proper mode codes such as, for example, `chmod 755 pdf2pdfa`.

Now the user must simply issue on the terminal the command:

```
pdf2pdfa myXeLaTeXfile
```

and ghostscript will do the whole work.

Of course the last step is to check the PDF/A compliance with a reliable program, such as Preflight. Please do not trust the information issued by Adobe Reader X when it opens a PDF file that pretends to be PDF/A compliant. The information is a wish, in the sense that the statement might be true, but the document has not been really tested in every remote corner of its compliance. So the real message should not be: "This file is PDF/A compliant", but:"This file might be PDF/A compliant".

5 Conclusion

I have converted a number of PDF documents produced with XeLaTeX using the procedure described in the previous sections; of course I have observed all the caveats I listed above; actually they are the result of my failing efforts; I had to find out at every failure the cause, but eventually I could put together a reasonable list of caveats.

There is another thing that might be done: to write an extension file to be read by the main document file, that accepts LaTeX style commands in order to specify the *metadata* and write the auxiliary file without any specific intervention by the user; this extension should also run ghostscript by means of a suitable "shell escape" command at the end of the processing by XeLaTeX, as the last task of the `\end{document}` statement. Apparently the necessary hooks already exist, but I must leave this further task for another moment.

References

Adobe (2006). «Adobe's free xmp toolkit for reading/writing xmp». `http://www.adobe.com/devnet/xmp/sdk/eula.html`. In basso a

The auxiliary file

```
%!
% $Id: PDFA_def.ps 8284 2007-10-10 17:40:38Z giles $
% This is a sample prefix file for creating a PDF/A document.
% Feel free to modify entries marked with "Customize".

% This assumes an ICC profile to reside in the file (ISO Coated sb.icc),
% unless the user modifies the corresponding line below.


systemdict /ProcessColorModel known {
  systemdict /ProcessColorModel get dup /DeviceGray ne exch /DeviceCMYK ne and
} {
  true
} ifelse
{ (ERROR: ProcessColorModel must be /DeviceGray or DeviceCMYK.)=
  /ProcessColorModel cvx /rangecheck signalerror
} if


% Define entries to the document Info dictionary:

/ICCProfile (/Users/claudio/icc/ECI-RGB.V1.0.icc) def % Customize

[ /Title (XeLaTeX and the PDF archivable format) % Customize.
/Author (Claudio Beccari) % Customize.
/Subject (How to produce a PDF/A compliant document typeset with XeLaTeX) % Customize.
  /DOCINFO pdfmark

% Define an ICC profile :

[/_objdef {icc_PDFA} /type /stream /OBJ pdfmark
[{icc_PDFA} <</N systemdict /ProcessColorModel get
          /DeviceGray eq {1} {4} ifelse >> /PUT pdfmark
[{icc_PDFA} ICCProfile (r) file /PUT pdfmark

% Define the output intent dictionary :

[/_objdef {OutputIntent_PDFA} /type /dict /OBJ pdfmark
[{OutputIntent_PDFA} <<
  /Type /OutputIntent             % Must be so (the standard requires).
  /S /GTS_PDFA1                   % Must be so (the standard requires).
  /DestOutputProfile {icc_PDFA}            % Must be so (see above).
  /OutputConditionIdentifier (CGATS TR001)      % Customize
>> /PUT pdfmark
[{Catalog} <</OutputIntents [ {OutputIntent_PDFA} ]>> /PUT pdfmark
```

The bash script

```
#!/bin/bash
file1=$1.pdf
file2=$1-a.pdf
file3=$1-def.ps
# WHAT FOLLOWS MUS BE WRITTEN IN JUST ONE LINE
gs -dPDFA -dNOOUTERSAVE -dUseCIEColor -dCompatibilityLevel=1.4 -sDEVICE=pdfwrite
   -sProcessColorModel=DeviceCMYK -sPDFACompatibilityPolicy=1 -o $file2 ./$file3 $file1
```

sinistra nella schermata puntata dall'URL c'è l'ancora con il link per il download.

Beccari, C. (2010). «Some PDF/A tricks». *The PracTEX Journal*.

Drümmer, O., Oettler, A. e von Seggern, D. (2008). *PDF/A in a Nutshell. Long Term Archiving with PDF*. Callas Software gmbh. In: http://www.pdfa.org/doku.php?id=pdfa:en:pdfa_in_a_nutshell.

Hàn Thế Thành (2008). «Generating PDFA compliant PDFs from pdftex». http://support.river-valley.com/wiki/index.php?title=Generating_PDF/A_compliant_PDFs_from_pdftex.

Knuth, D. E. (1996). *The TEXbook*. Addison Wesley, Reading, Mass., 16ª edizione.

Martini, E. (2007). «Lo standard PDF/A: Sperimentazione di software per la verifica di conformità allo standard ISO 19005:2005». http://www.cnipa.gov.it/site/_files/ref02_RS_3.1_martini.pdf.

PDF/A (2005–2008). «Technical notes». http://www.pdfa.org/. This site contains the Technical Notes included in the following files: tn0001_pdfa-1_and_namespaces_2008-03-19.pdf, tn0003_metadata_in_pdfa-1_2008-03-18.pdf, tn0006_digital_signatures_in_pdfa-1_2008-03-14.pdf, tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf, tn0009_xmp_extension_schemas_in_pdfa-1_2008-03-20.pdf.

Radhakrishnan, C. e Hàn Thế Thành (2008). «Generation of PDF/X-1a and PDF/A-1b compliant PDF's with PDFTEX — pdfx.sty». http://tug.ctan.org/tex-archive/macros/latex/contrib/pdfx/.

Scarso, L. (2010). «Introduction to colours in ConTeXt mkiv». *TUGboat*, **31** (3), pp. 203–207.

▷ Claudio Beccari
Villarbasse (TO)
claudio dot beccari at gmail dot com

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 dei 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[3] 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}
```

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 dei grafici realizzati è disponibile su internet alla pagina `https://github.com/robitex/pgfplots-article/`.

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

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[4].

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 a partire da 0.

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*⟩.

4. 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.

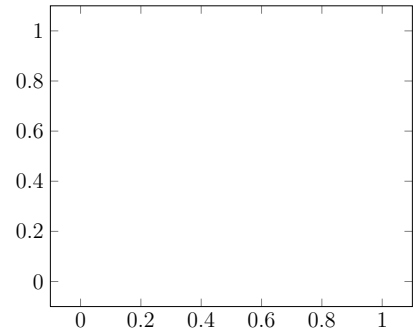
Nel codice precedente le chiavi xmin e xmax fissano il valore minimo e il valore massimo delle ascisse. Analoghe chiavi vengono adoperate per l'asse delle ordinate. L'utente deve tener presente che quando ha esigenze particolari sull'impostazione dei limiti di un asse (ad esempio delle ordinate) è tenuto 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.

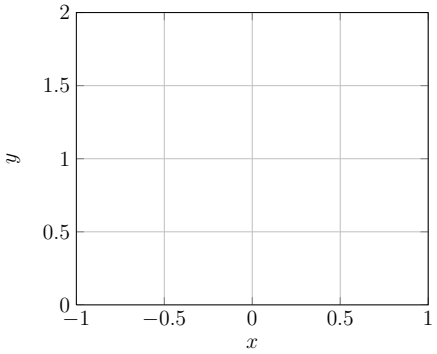
Le opzioni xlabel e ylabel consentono di inserire le etichette degli assi, in questo caso corrispondenti a x e y. Va osservato che l'impostazione predefinita riguardante l'etichetta dell'asse delle ordinate corrisponde alla prassi molto diffusa secondo la quale essa viene ruotata di 90° e centrata verticalmente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

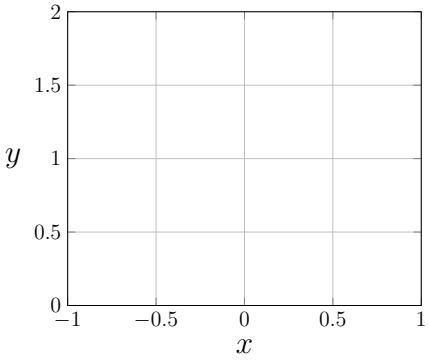
```
% nel preambolo
\usepackage{pgfplots,relsize}
...
% impostazione degli stili di pgfplots
\pgfplotsset{
    every axis x label/.append style = {
       font = \relsize{1}
    },
    every axis y label/.append style = {
       font = \relsize{1},
       rotate = -90,
       xshift = 0.5em
    }
}
\begin{tikzpicture}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



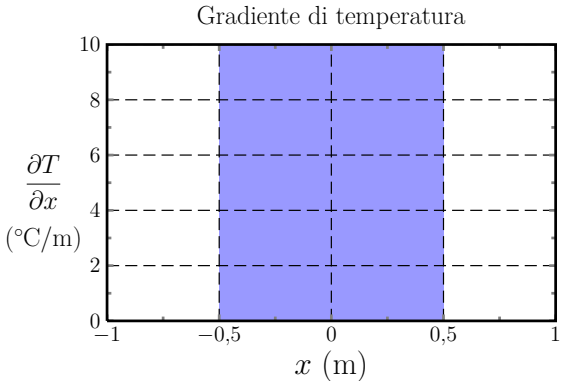
(a) Uso minimale dell'ambiente axis.



(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}
```

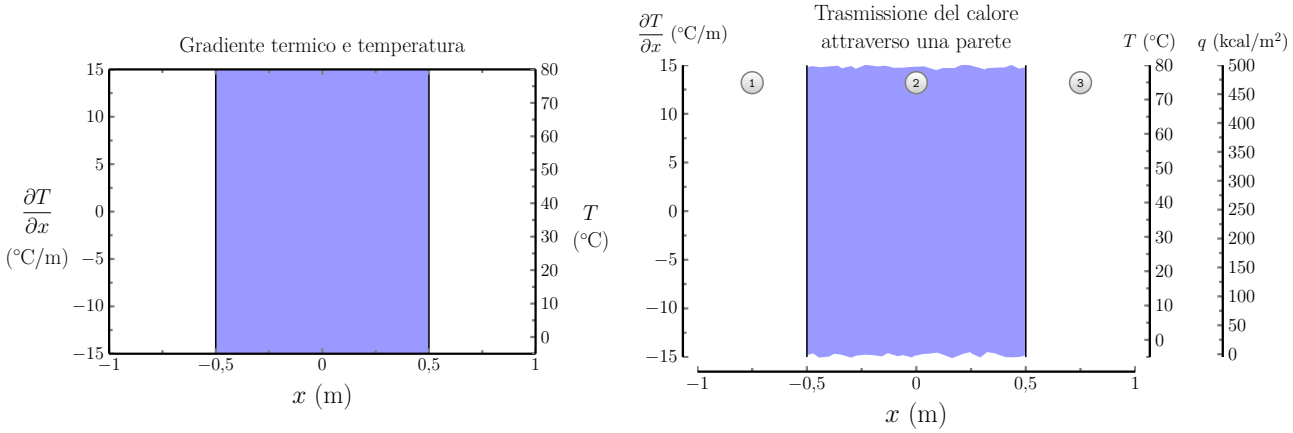
Le personalizzazioni introdotte nella figura 1d sono le seguenti:

(*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

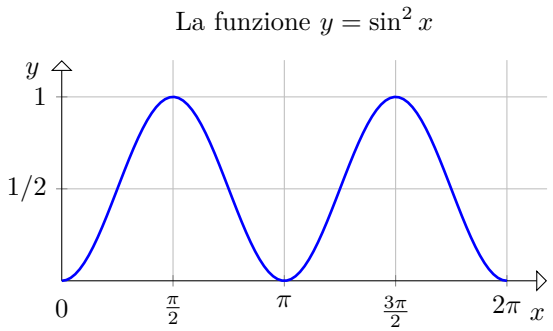


FIGURA 3: Esempio di studio di funzione

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 PGF si 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 ricordare 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 loro 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.

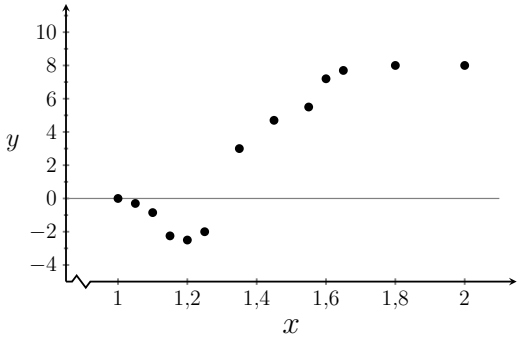
Il codice completo che genera il grafico della figura 3 è il seguente:

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

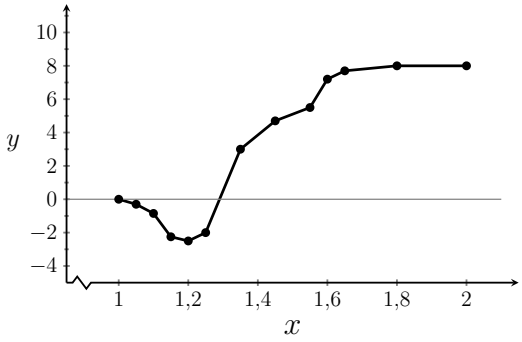
```
      xmax=6.85,
      ymin=0,
      ymax=1.20,
      axis x line=bottom, % posizione assi
      axis y line=left,
      % tipo di freccia
      every outer x axis line/.append style=
         {-open triangle 90},
      every outer y axis line/.append style=
         {-open triangle 90},
      % 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.

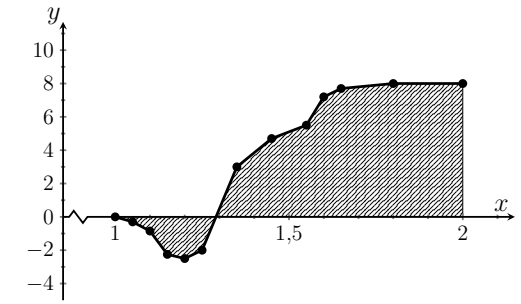
Si osservi che PGFPLOTS aiuta a trovare le coordinate dei punti nel grafico (dentro la tela o al di fuori di essa) mettendo a disposizione un certo numero di sistemi di riferimento appositamente definiti. I più usati sono individuati dai nomi: `axis cs` e `axis description cs`. È buona norma, quando 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

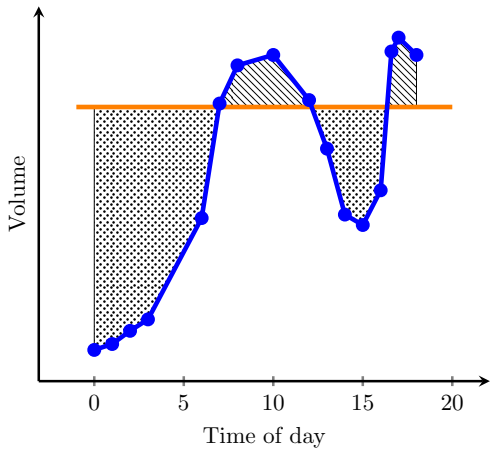


FIGURA 5: Esempio avanzato in cui si tratteggiano le varie regioni comprese fra una spezzata ed una retta orizzontale diversa dall'asse delle ascisse.

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[5].

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 scontornando una pagina con `pdfcrop`.

5. 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`.

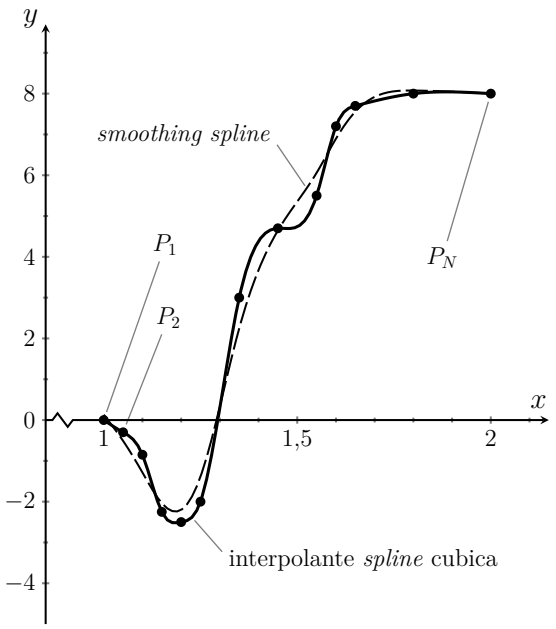


FIGURA 6: Grafico di funzioni interpolanti. I punti di supporto (pallini) sono i medesimi della figura 4.

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. Oltre ad avere molte caratteristiche specifiche del loro campo di applicazione, questi programmi hanno la capacità di ricercare 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-

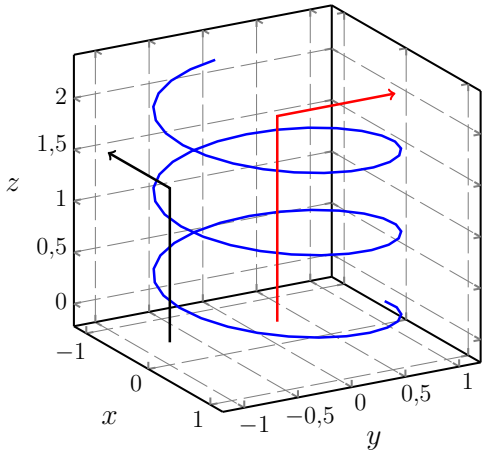


FIGURA 7: Un'elica disegnata con `\addplot3`.

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`[6] 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

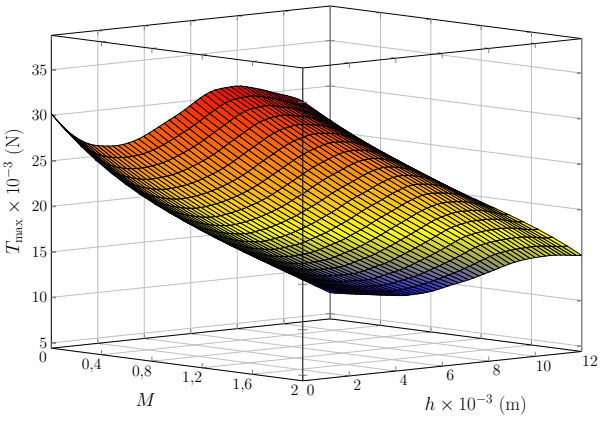


FIGURA 8: Superficie prodotta con l'ausilio di `matlab2tikz.m`.

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

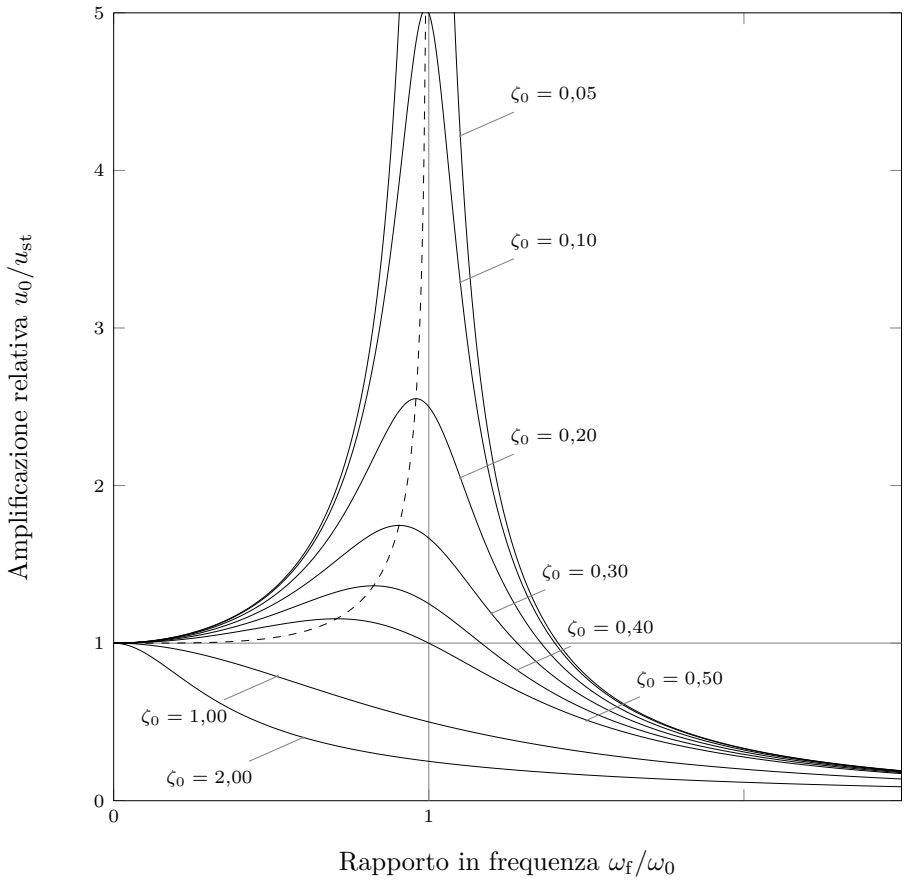


Figura 9: La famiglia di curve di risonanza dell'oscillatore semplice.

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, caratterizzato da 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 tacche di marcatura).

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[7].

```
\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 esigenza di chiarezza — è possibile evitare di apporre una marcatura "per tentativi"[8]. 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 $(\Delta x,\Delta y)$ 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

7. 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.

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

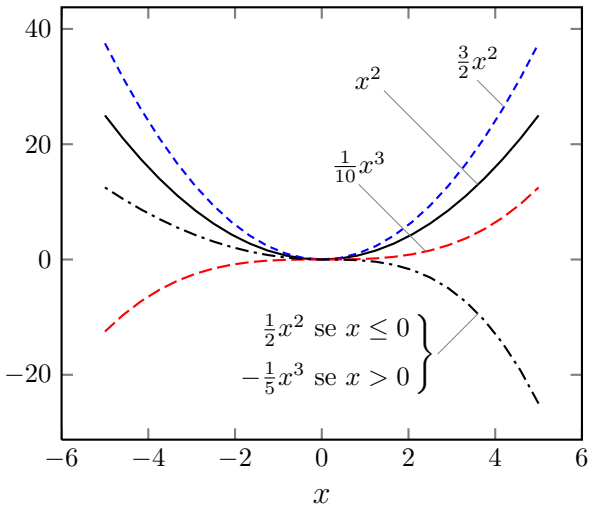


FIGURA 10: Curve multiple etichettate direttamente attraverso la definizione di un nuovo `tikzstyle`.

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
      $
        left.\begin{array}{rl}
          \frac{1}{2}x^2 &\text{se } x \le 0\\
          -\frac{1}{5}x^3 &\text{se } x > 0
        \end{array}\right\}
      $
    }
  ] {(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`.

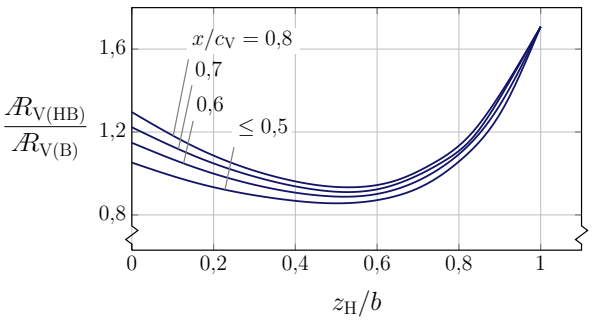


FIGURA 11: Un altro esempio di grafico in cui l'etichettatura delle curve è stata ottenuta in maniera semiautomatica.

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}[
   % ...
```

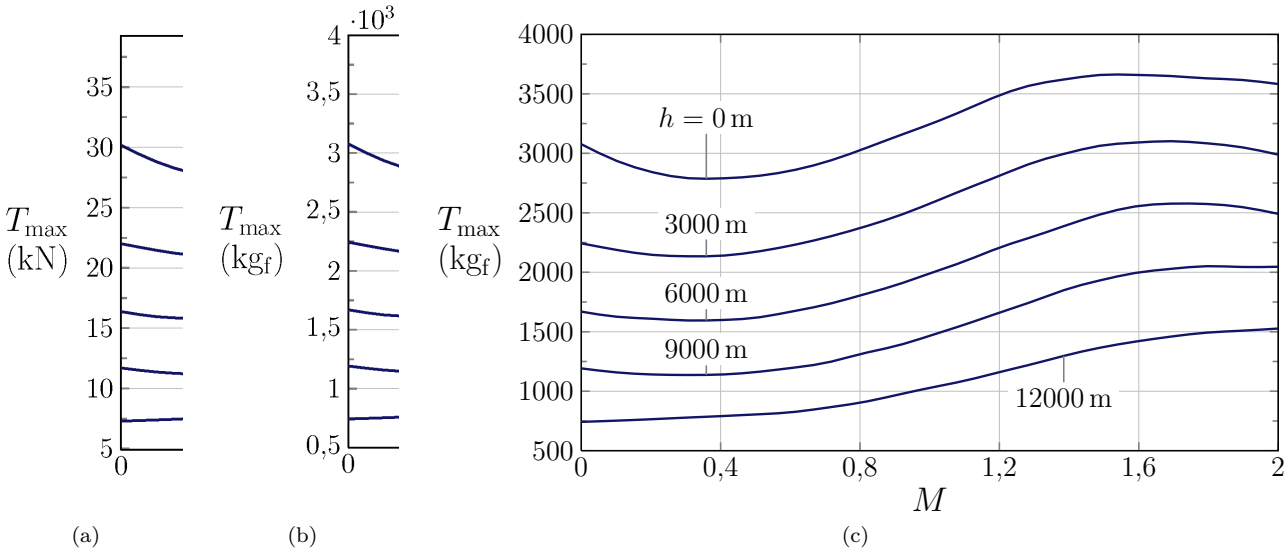


FIGURA 12: Regolare i valori delle etichette numeriche.

```
    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 dal 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,
```

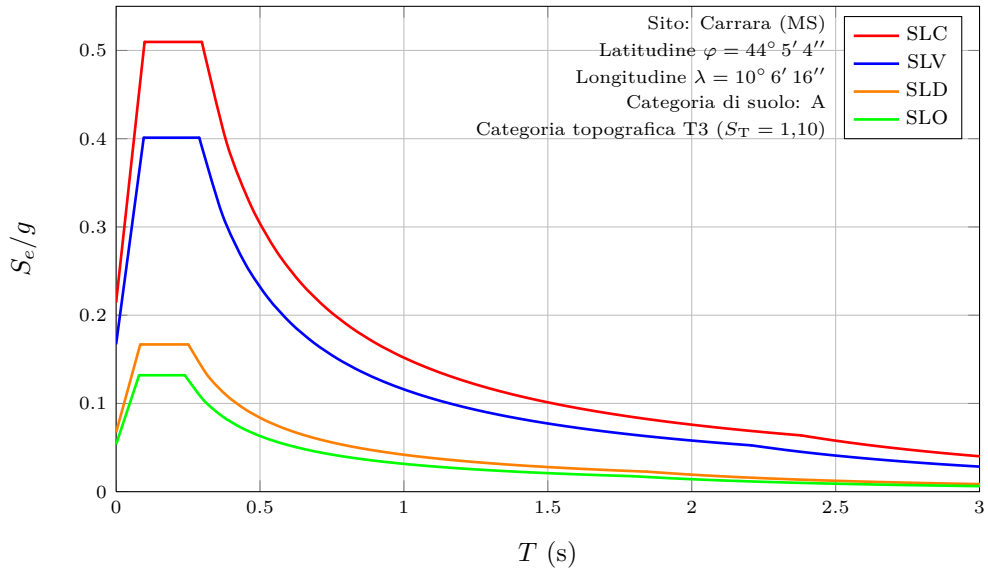


FIGURA 13: Spettri di risposta come curve a più rami

```
    % 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
```

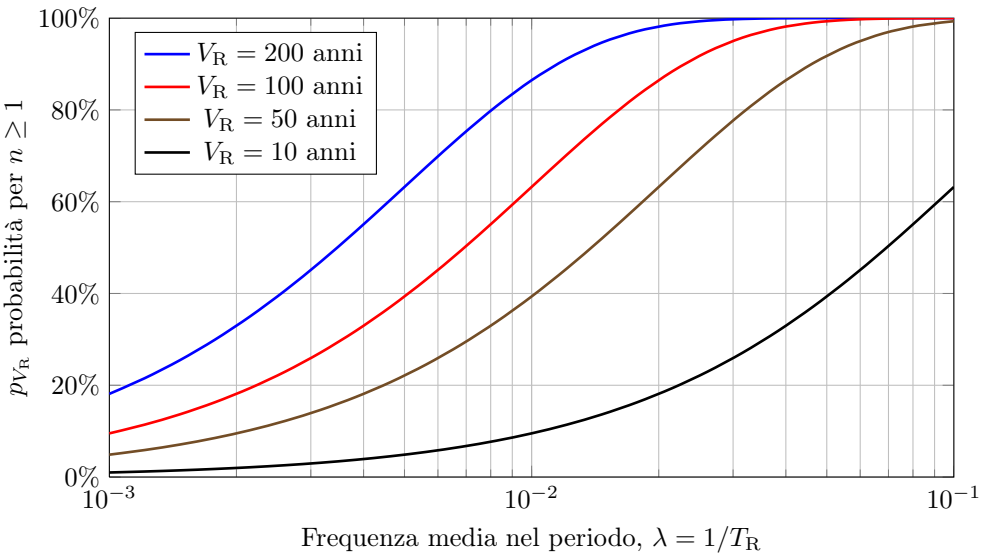


FIGURA 14: Un tipo di grafico semilogaritmico in ascissa

```
          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[9] 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[10].

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]
```

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

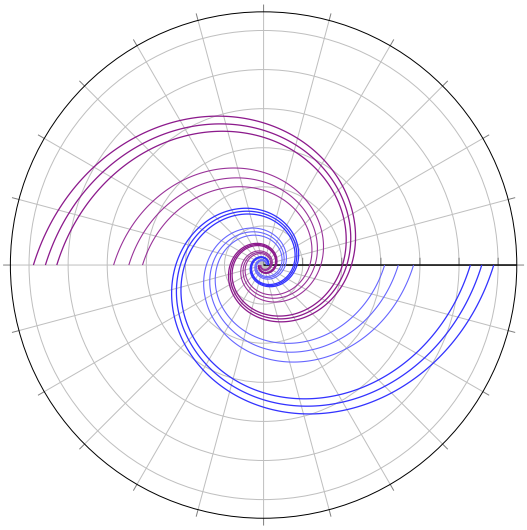


FIGURA 15: Intreccio galattico: una serie di spirali logaritmiche *auree* nel piano polare di PGFPLOTS

```
        {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 somigliante 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})},
     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}
```

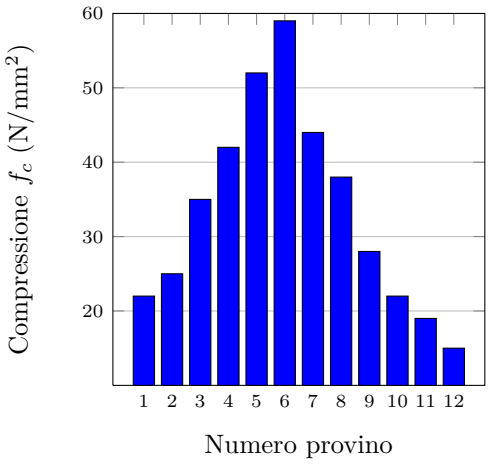


FIGURA 16: Un esempio di un istogramma

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

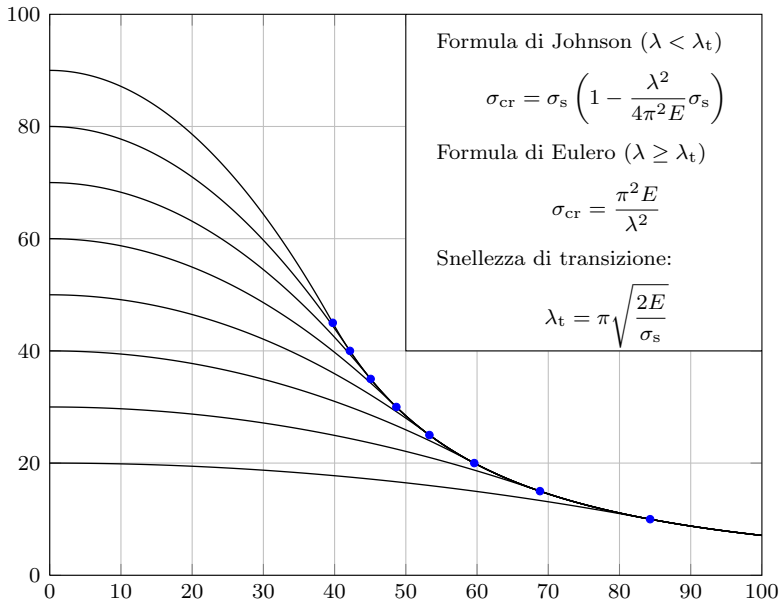


FIGURA 17: Curve di Euler-Johnson da file dati costruiti in Lua

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[11] che utilizzeremo per sperimentare la procedura proviene dalla Scienza delle costruzioni. Ci rifaremo alla stabilità dell'equilibrio di una trave sollecitata con un carico assiale, in particolare alla Teoria 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 in 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, in Java ed in molti altri linguaggi di programmazione, le stringhe sono tipi *immutabili*

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

cioè non possono essere modificate. La concatenazione di due stringhe non provoca l'aggiornamento di un elemento esistente ma la creazione di una nuova variabile trasferendo entrambi i blocchi di memoria. Se in un ciclo si concatenassero ad una stessa variabile 1000 stringhe di 10 caratteri ciascuna per soli 10 kB di dati, alla fine si sarebbero movimentati in memoria oltre 5 MB di dati.

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 la definizione 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}
```

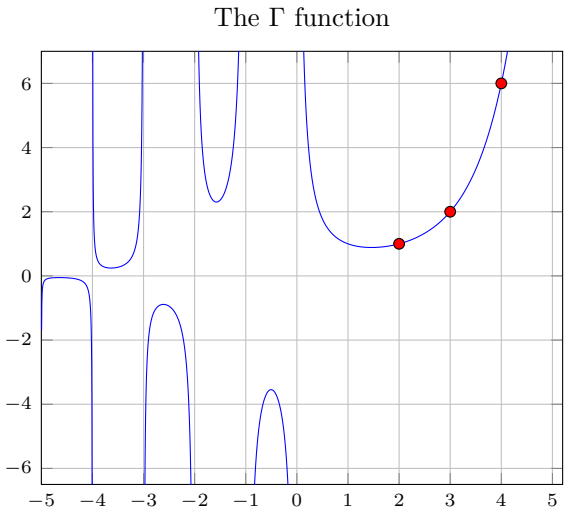


FIGURA 18: La funzione Γ di Eulero

```
\end{tikzpicture}
\end{document}
```

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'indentazione 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[12], 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.

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.

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

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`

LaTeX nella pubblica amministrazione. La web application FACILE per la produzione automatica di comunicazioni interne del Comune di Napoli

Agostino De Marco, Paolo Eugenio Cresci

Sommario

In questo articolo viene presentata la web application FACILE in uso presso l'Amministrazione Comunale di Napoli. L'applicazione è stata rilasciata nel marzo 2011. Essa è usabile attraverso la intranet del Comune di Napoli ed è attualmente in fase di sperimentazione. Gli utenti sono in grado di produrre comunicazioni ufficiali ad uso interno secondo le specifiche del manuale della *Corporate Identity* del Comune di Napoli. L'applicazione è pensata per essere di semplice utilizzo e presenta all'utente una *web form* opportunamente progettata per la produzione di una lettera in formato PDF. Dietro le quinte FACILE utilizza il motore di composizione XeLaTeX ed un *parser* sviluppato in linguaggio PHP. L'aspetto chiave è la progettazione di un *template* di sorgente LaTeX basato sulla classe scrlttr2 appartenente al *bundle* KOMA-script.

Abstract

The article introduces the web application FACILE, in use at the Administration of the City of Naples. The application was released in March 2011. It is integrated into the intranet software tools of the Municipality and is currently under testing. Users are able to produce official internal communication letters which comply to the City of Naples Corporate Identity specifications. The application is designed to be perceived as a very simple tool and presents itself to the user as a web form for the production of a PDF document. Behind the scenes FACILE runs the XeLaTeX typesetting engine together with an *ad hoc* parser developed in the PHP language. The key aspect of the application is the design of a template of LaTeX source based on the scrlttr2 document class from the KOMA-script bundle.

1 Introduzione

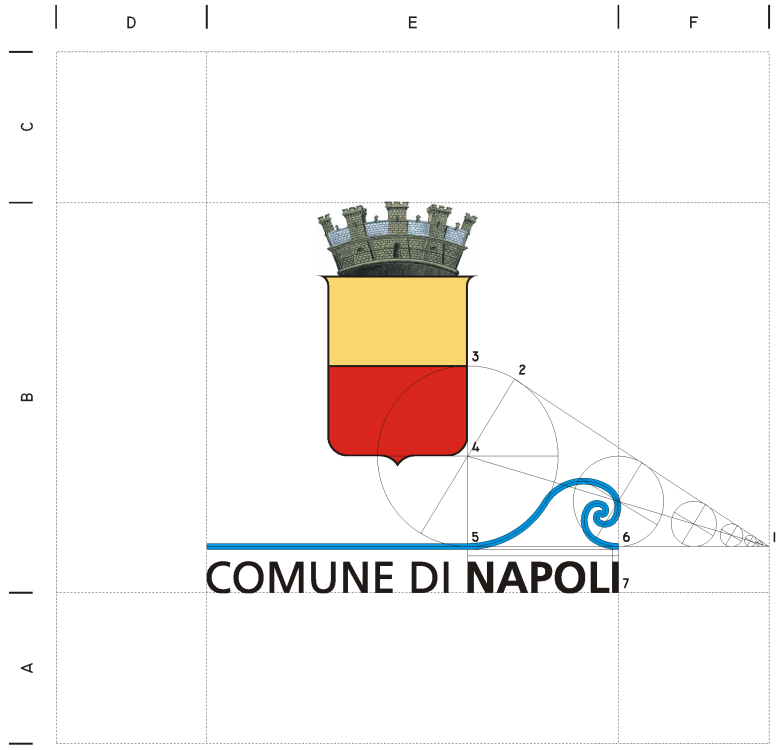
Negli ultimi anni la necessità di abbattimento dei costi aziendali, tanto nel settore pubblico quanto in quello privato, ha fatto emergere un crescente interesse verso il mondo del software libero e *open source*. L'Amministrazione comunale di Napoli è un esempio di ente pubblico che ha saputo mostrare buone intuizioni nel campo dell'ottimizzazione dei sistemi informativi. Alla fine del 2007, con la delibera n. 3999 quale atto di indirizzo in tema di sviluppo tecnologico e centralizzazione degli acquisti, la Giunta comunale di Napoli indica le modalità per la graduale migrazione verso l'uso di tecnologie a codice aperto: (*i*) azioni pilota che dimostrassero l'applicabilità del percorso di sviluppo; (*ii*) priorità nella predisposizione e partecipazione a progetti cofinanziati con fondi europei, nazionali e regionali previsti per la diffusione dell'uso del codice aperto; (*iii*) specifici e differenziati percorsi formativi del personale, nell'ambito della formazione annualmente erogata dall'Ente. Nel 2009 gli organi dirigenti dell'Ente hanno disposto la formazione di un Gruppo di lavoro interservizi denominato "Ricambio del parco microinformatico del Comune di Napoli e passaggio generalizzato all'*open source*". Questa iniziativa ha portato al rinnovamento di una parte delle postazioni informatiche dislocate nei vari uffici sul territorio, arrivando in un anno a sostituire 1800 vecchi computer a noleggio con altrettante postazioni di nuova fornitura, dotate di sistema operativo Linux, collegate in rete e attrezzate per le diverse necessità dei vari dipartimenti.

Tra le altre iniziative recenti degne di nota vi sono: il patrocinio da parte del Comune di Napoli degli ultimi due *Linux Day* e l'istituzione a marzo del 2011 di un Osservatorio Campano sull'Open Source. Quest'ultimo, indipendentemente dai risultati concreti che esso porterà e che saranno verificabili solo nel futuro, porta in sé un'idea interessante: un nucleo di valutazione istituito da una pubblica amministrazione composto da servizi tecnici, consiglieri comunali, ricercatori esperti di comunicazione e associazioni che promuovono il software libero, che ha il compito di giudicare l'efficacia della migrazione verso il software libero e *open source* e promuovere lo stesso processo per quel che riguarda i cosiddetti "applicativi verticali" utilizzati negli uffici dell'Ente (*database*, software di gestione dei *web server*, eccetera); un gruppo di lavoro che valuta anche le eventuali richieste d'acquisto dei software commerciali da parte dei servizi tecnici e amministrativi del Comune.

Con questa premessa, nelle pagine che seguono si cerca di mostrare come sia possibile sposare le

(a) Versione attuale, elaborata nel 2006, dello stemma araldico, dell'onda e del logotipo.

(b) Versione monocromatica adatta alla riproduzione su carta tramite punzonatura.



(c) Prescrizione delle distanze di rispetto per il corretto posizionamento dello stemma all'interno di manifesti, documenti ufficiali e altro materiale editoriale.

Figura 1: Il marchio del Comune di Napoli.

attuali possibilità offerte dal software libero e aperto con le esigenze della comunicazione istituzionale all'interno di una pubblica amministrazione. In particolare, in alternativa all'uso di programmi *open source* dedicati al *desktop publishing*, si racconteranno i motivi che hanno portato a scegliere una soluzione basata sul motore di composizione XeLaTeX sperimentando nuovi modi di operare nel campo dell'*office automation*. L'esperimento è tanto più interessante se si pensa che la soluzione applicativa che viene presentata in questo articolo è potenzialmente utilizzabile da migliaia di dipendenti comunali (circa 5000) che vi hanno accesso.

Nella prossima sezione si parlerà del progetto di identità visiva del Comune di Napoli e della sua applicazione negli strumenti di comunicazione istituzionale. In particolare, l'attenzione sarà focalizzata sul formato delle lettere di comunicazione ufficiali specificato dal manuale della *Corporate Identity* dell'Ente. Nella sezione successiva si presenterà la *web application* FACILE — sviluppata utilizzando software libero o aperto — realizzata nell'ambito del progetto ADMINISTRA ("gestione elettronica degli Atti e dei Documenti amMINISTRAtivi"), per la creazione guidata di lettere di comunicazione istituzionale ad uso interno ("FAi una LEttera secondo la *Corporate Identity*"). Nella parte finale dell'articolo saranno presentati alcuni aspetti tecnici (e TEXnici) legati all'integrazione dell'interfaccia utente di FACILE con il compilatore XeLaTeX e con un *template* di sorgente basato sulla classe scrlttr2.

2 L'identità visiva del Comune di Napoli (*Corporate Identity*)

Nel Novembre 2005, l'Amministrazione comunale di Napoli indice un concorso di idee per la realizzazione della linea grafica dell'Ente. Tra le linee guida del concorso viene stabilito di mantenere inalterato lo stemma della città, innovandolo attraverso un'elaborazione del simbolo stesso che ne migliorasse la leggibilità e la riproducibilità conservando immutati gli elementi distintivi.

L'antico stemma e la nuova onda sono i due inequivocabili segni del nuovo logo del Comune di Napoli mostrato nella figura 1. Il marchio sintetizza le idee di tradizione ed innovazione, di antico e moderno, tutti elementi caratterizzanti la citta di Napoli, e vuole comunicare all'osservatore la propensione a guardare al futuro senza distogliere lo sguardo dal passato, fatto di storia, cultura e tradizioni.

Il nuovo logo del Comune di Napoli ha rappresentato il punto di partenza di un progetto volto a dare alla città un vero e proprio *strumento di identificazione*, e, nel contempo, a definire la *Corporate Identity* dell'Amministrazione comunale. Nelle

Tabella 1: Tipi di carte da lettera istituzionale, definita nella sezione "Modulistica Istituzionale" del manuale della *corporate identity* del Comune di Napoli.

Nome	*Mittente*	*Formato*	*Testi* (composizione intestazioni, piedini, eccetera)	*Colori* (marchio e testi)	*Materiale Carta*
Tipo A1	Il Sindaco e i suoi collaboratori (il marchio prevede la variante *senza onda* con il logotipo "Il Sindaco di Napoli")	UNI A4	Adobe Garamond corsivo e nero-corsivo, Frutiger nero	marchio in colori metallici; testi in colore nero.	tipo Fedrigoni Splendorgel Extra White, di peso non inferiore a 80 g/m^2
Tipo A2	Il Sindaco e i suoi collaboratori (il marchio prevede la variante *senza onda* con il logotipo "Il Sindaco di Napoli")	UNI A4	Adobe Garamond corsivo e nero-corsivo, Frutiger nero	marchio in colori metallici; testi in colore nero.	tipo Fedrigoni Corolla Classic Ivory, di peso non inferiore a 80 g/m^2
Tipo A3	Il Sindaco e i suoi collaboratori (il marchio prevede la variante *senza onda* con il logotipo "Il Sindaco di Napoli" ed è punzonato a secco)	UNI A4	Adobe Garamond corsivo e nero-corsivo, Frutiger nero	marchio in colori metallici; testi in colore nero.	tipo Fedrigoni Corolla Classic Premium White, di peso non inferiore a 80 g/m^2
Tipo B	Vice Sindaco, Assessori, Direttore Generale, Segretario Generale, Capo di Gabinetto, Vice Segretario Generale, Presidente e Vice Presidenti del Consiglio Comunale, Consiglieri Comunali, Presidenti di Commissione, Gruppi Consiliari	UNI A4	Adobe Garamond corsivo e nero-corsivo, Frutiger nero	marchio in ciano, magenta, giallo e nero; testi in colore nero.	tipo Fedrigoni Splendorgel Extra White, di peso non inferiore a 80 g/m^2
Tipo C	Direzioni Centrali, Dipartimenti Autonomi, Servizi Autonomi	UNI A4	Adobe Garamond corsivo e nero-corsivo, Frutiger nero	marchio in ciano, magenta, giallo e nero; testi in colore nero.	tipo Fedrigoni Splendorgel Extra White, di peso non inferiore a 80 g/m^2
Tipo D1	Presidente della Municipalità	UNI A4	Adobe Garamond corsivo e nero-corsivo, Frutiger nero	marchio in ciano, magenta, giallo e nero; testi in colore nero.	tipo Fedrigoni Splendorgel Extra White, di peso non inferiore a 80 g/m^2
Tipo D2	Altri organi di governo della Municipalità.	UNI A4	Adobe Garamond corsivo e nero-corsivo, Frutiger nero	marchio in ciano, magenta, giallo e nero; testi in colore nero.	tipo Fedrigoni Splendorgel Extra White, di peso non inferiore a 80 g/m^2

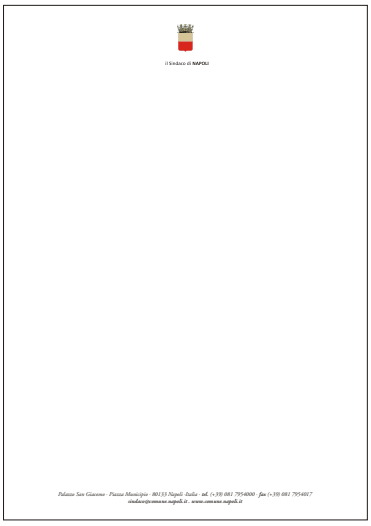
scienze della comunicazione la *corporate identity* ("identità societaria" o anche "identità visiva") è intesa come una strategia condivisa che unifichi per l'intera azienda il modello comunicativo, sia al suo interno che verso l'esterno.

L'idea alla base dell'attuale identità visiva del Comune di Napoli è quella di difendere, aggiornandola, l'identità di una città, facendone strumento di proiezione in ambito euromediterraneo. Il progetto grafico si è mosso su due direttrici: (*i*) il recupero dello stemma della città, con un intervento mirato ad assicurarne la massima leggibilità, l'adozione del logotipo "comune di napoli" e l'utilizzo dell'onda come segno; (*ii*) la definizione e l'unificazione dei materiali di comunicazione dell'azienda comunale. È proprio quest'ultimo punto il motivo ispiratore del presente articolo.

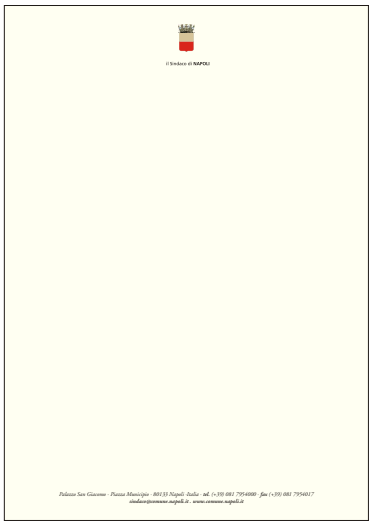
2.1 Il manuale della *Corporate Identity* del Comune di Napoli

La creazione della *corporate identity* dell'Ente risale al 2006 ed è documentata in un apposito manuale (Anonimo, 2006) che presenta gli elementi di identità visiva del Comune di Napoli. Il suo scopo è quello di fornire una guida agli addetti ai lavori — dipendenti e consulenti — affinché si superi la frammentazione nei materiali e nel segno grafico riscontrata nel lavoro di uffici e assessorati e si operi secondo un'unica strategia comunicativa, approvata ed adottata dall'intero Ente.

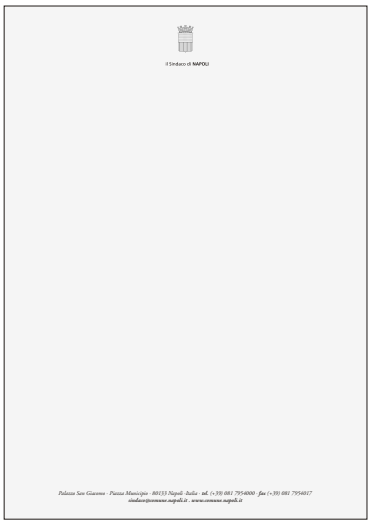
La linea grafica del Comune di Napoli, e la sua applicazione nei diversi strumenti di comunicazione interna ed esterna all'Ente, *deve* essere rigorosamente rispettata, così come definita nel manuale, senza possibilità di adattamenti e interpretazioni personali. Chiunque debba operare oggi — dipendenti comunali o consulenti che si accingono a preparare lettere, manifesti o altre forme ufficiali di comunicazione visiva — su un ambito, anche minimo, del sistema stesso, trova nelle tavole del manuale gli strumenti adatti. Il manuale è uno strumento nel quale cercare gli elementi che servono, trovandoli direttamente o cercando nella stessa famiglia la soluzione allo specifico problema.



(a) Lettera del Sindaco (Tipo A1).



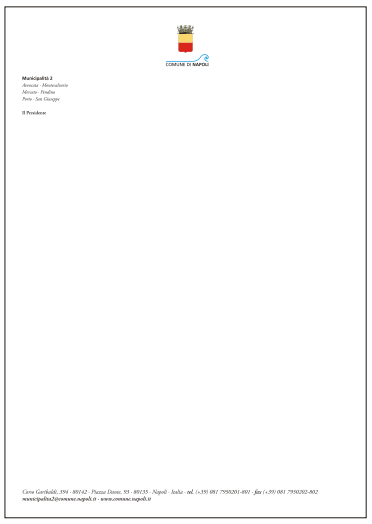
(b) Lettera del Sindaco (Tipo A2).



(c) Lettera del Sindaco (Tipo A3).

(d) Lettera dell'Assessorato ai Servizi Interni, Toponomastica e Censimenti (Tipo B).

(e) Lettera della Direzione Centrale I, Risorse Strategiche e Programmazione Economico-Finanziaria (Tipo C).



(f) Lettera del Presidente della Municipalità 2 (Tipo D).

FIGURA 2: Esempi di carta da lettera per comunicazioni ufficiali del Comune di Napoli (si veda anche la tabella 1). Testatina e piedini composti diversamente a seconda del tipo di comunicazione.

2.2 Lo stemma del Comune di Napoli

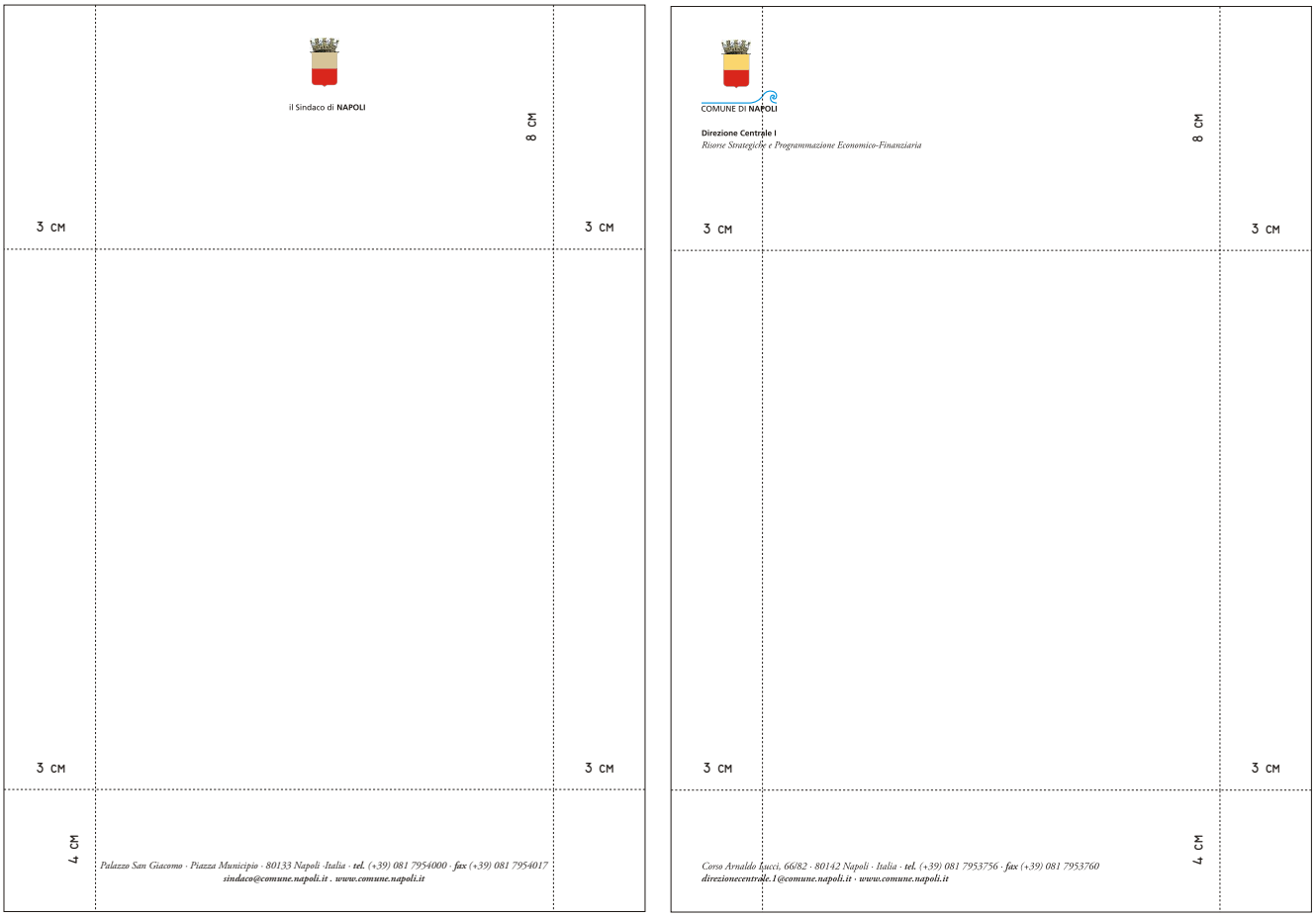
Il marchio rappresentativo del Comune di Napoli è costituito da tre elementi, combinati tra loro secondo un preciso schema grafico. Questi elementi sono: (*i*) l'antico stemma; (*ii*) il nuovo simbolo, l'onda; (*iii*) il logotipo "COMUNE DI NAPOLI".

L'elemento più importante e caratterizzante è lo stemma, risalente al XIV secolo[1]. Una rappresentazione dello stemma è mostrata nella figura 1. Nella figura 1a si riporta la versione ufficiale, a colori. Nella figura 1b viene mostrata una variante monocromatica destinata a particolari utilizzi. La posizione e la distanza tra gli elementi che compongono il marchio è fissa e invariabile. Nella figura 1c sono mostrate le distanze di rispetto prescritte nel manuale dell'identità visiva per il corretto posizionamento dello stemma all'interno di manifesti, documenti ufficiali e altro materiale editoriale.

Il marchio, in tutte le versioni d'uso, non può essere riprodotto con una misura di base inferiore a 24,7 mm (la misura fa riferimento alla larghezza del logotipo). Al di sotto di tale misura gli elementi che compongono il marchio perdono infatti di leggibilità.

Per quanto riguarda il cosiddetto *lettering* istituzionale, come carattere del marchio è stato scelto il font Frutiger (un font senza grazie), negli stili tondo e nero. Nel logotipo la parte "COMUNE DI" è in tondo e la parte "NAPOLI" è in nero. Come estensione consentita del carattere istituzionale è possibile utilizzare unicamente il font Frutiger

1. Si vedano: `http://it.wikipedia.org/wiki/Stemma_di_Napoli` ed `http://www.comune.napoli.it/flex/cm/pages/ServeBLOB.php/L/IT/IDPagina/3504`.



(a) Lettera del Sindaco (Tipo A1).

(b) Lettera da parte di Direzioni Centrali, Dipartimenti Autonomi, Servizi Autonomi (Tipo C).

FIGURA 3: Modelli di carta da lettera per comunicazioni ufficiali del Comune di Napoli. Definizione della gabbia del testo (si veda anche la tabella 1).

nello stile corsivo. Non è consentito l'utilizzo di altri caratteri. Per garantire una buona leggibilità è stato individuato il corpo 7,5 come dimensione minima, in riferimento al formato UNI A4, cioè il formato di carta maggiormente usato.

2.3 Modelli di carta da lettera

Il manuale dell'identità visiva dell'Ente definisce in maniera sistematica i formati ammissibili delle comunicazioni ufficiali, destinate sia alla stampa che all'archiviazione elettronica. Il recapito delle comunicazioni, a seconda della tipologia e dei destinatari, in genere viene effettuato per via postale oppure per via elettronica.

Le comunicazioni istituzionali possono avere uno o più destinatari ('principali' e 'in copia') e si distinguono in due categorie: le comunicazioni interne e le comunicazioni esterne. Le comunicazioni interne sono destinate ad uso interno dell'Ente e riguardano tipicamente comunicazioni tra dipartimenti, uffici, eccetera. Le comunicazioni esterne sono dirette da un soggetto istituzionale appartenente all'Amministrazione comunale a un destinatario esterno all'Ente, istituzionale anch'esso o privato.

Il quadro dei tipi di lettera istituzionale definiti nel manuale della Corporate Identity del Comune di Napoli è riportato nella tabella 1. Alcuni esempi di lettera — un foglio bianco con testatina e piedini composti diversamente a seconda del tipo di comunicazione — sono mostrati nella figura 2. Nella figura 3 sono riportati due esempi tratti dal manuale dell'identità visiva in cui sono specificate nel dettaglio le dimensioni dei margini che individuano la gabbia del testo.

Le categorie di comunicazione vengono definite in base al mittente (Tipo A1, A2 A3, B, C, D1, D3). L'applicazione FACILE (che sarà descritta più avanti) è stata progettata in base all'esigenza iniziale di uniformare i formati delle comunicazioni *interne*. Essa introduce una procedura guidata attraverso un'interfaccia utente (una *web form*) che consente la creazione di lettere istituzionali di Tipo C con destinatari interni all'ente. Questa limitazione è dovuta al carattere innovativo e sperimentale dell'applicazione. Si è voluto restringere il campo dei possibili tipi di documento da produrre in maniera automatica, per poi passare ad una fase successiva di 'sperimentazione sul campo' ed infine valutare

FIGURA 4: La pagina iniziale della intranet del Comune di Napoli. L'applicazione FACILE è integrata nel menu a disposizine dei dipendenti.

l'efficacia dello strumento di lavoro.

La modulistica di Tipo C è prevista per le comunicazioni interna tra Direzioni Centrali, i Dipartimenti Autonomi e i Servizi Autonomi del Comune. L'impostazione grafica prevede il posizionamento a bandiera a sinistra del marchio ed una distribuzione a bandiera a sinistra dei testi all'interno della testatina e del piedino. La comunicazione deve avere un formato di pagina di tipo UNI A4. Il font stabilito per il testo nella testatina e nel piedino è Adobe Garamond (corsivo e nero-corsivo) e Frutiger (tondo e nero). Il font del testo della comunicazione è Adobe Garamond. Per garantire una buona leggibilità è stato individuato il corpo 10. Non è consentito l'utilizzo di altre famiglie di caratteri. La riproduzione del marchio corrisponde a quella riportata nella figura 1a. Per le versioni a stampa deve essere usata carta di tipo Fedrigoni Splendorgel Extra White, di peso non inferiore a $80\,\mathrm{g/m^2}$, idonea ad essere stampata a più colori con tecnologia *offset* oppure dalle comuni stampanti laser.

Per quel che concerne l'impostazione della gabbia di testo nella pagina, per una lettera di Tipo C sono da rispettare i seguenti margini: un margine superiore di 8 cm, un margine inferiore di 4 cm, un margine destro di 3 cm, un margine sinistro di 3 cm. In generale i fogli delle lettere più lunghe di una pagina portano il testo solo sulle pagine dispari (*recto*).

In casi eccezionali, il sistema di comunicazione del Comune di Napoli, accanto ai caratteri Frutiger e Adobe Garamond, prevede una possibile estensione del *lettering* istituzionale. I font ufficiali possono essere sostituiti con i seguenti font compatibili: Verdana per quel che riguarda l'estensione del *lettering* del marchio, Verdana e Times New Roman per l'estensione del *lettering* della modulistica.

3 La web application FACILE

L'idea che ha ispirato il progetto di FACILE era rivolta al dipendente comunale addetto alla preparazione e all'invio di una comunicazione interna protocollata, invogliandolo all'uso di uno strumento diverso da un programma di video scrittura. Una motivazione iniziale era costituita dalla volontà di evitare che il singolo utente prendesse iniziative personali sul modo in cui comporre una lettera — usando eventualmente font non consentiti o vecchie versioni del logo dell'Ente oppure, ancora, riciclando il formato di vecchi documenti non aderenti alle specifiche del manuale della *corporate identity*.

In secondo luogo, considerato che un lavoratore comunale normalmente usa e gestisce la posta attraverso un *browser* collegato alla Intranet della pubblica amministrazione, è stato ritenuto un aspetto molto importante per la fortuna di questa sperimentazione la necessità di dare all'utente la percezione che la composizione di una comunicazione interna fosse tanto semplice quanto gestire la posta elettronica.

Pertanto la scelta progettuale è caduta su uno strumento di lavoro simile a una *webmail*, con la differenza che il risultato finale sarebbe dovuto essere un file in formato PDF da stampare (e successivamente protocollare), archiviare e diramare. Data la

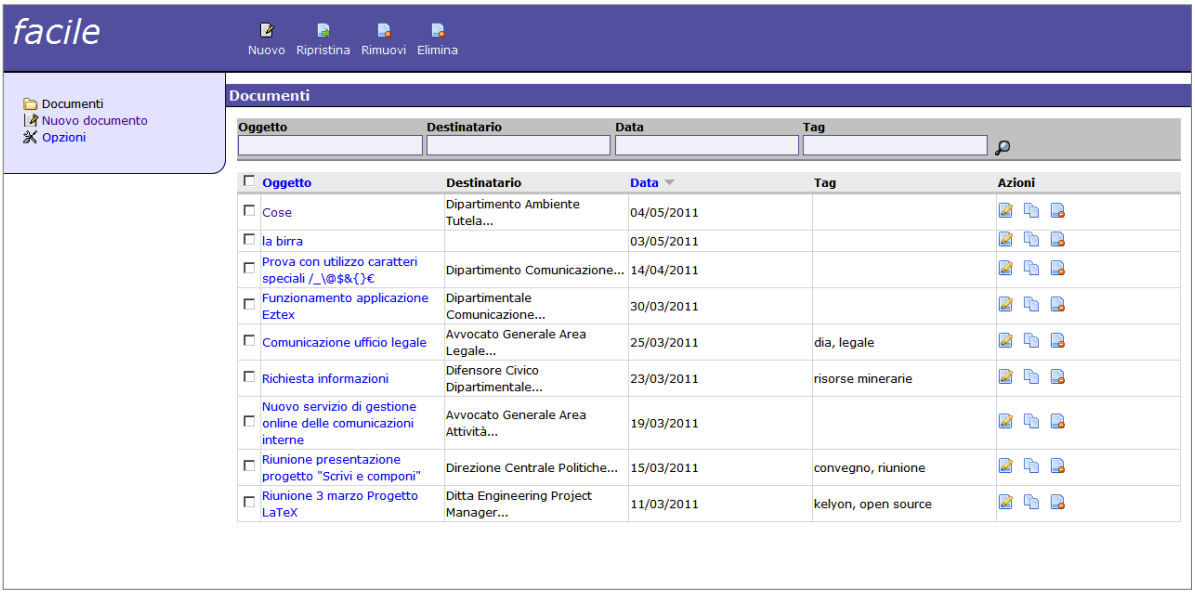


FIGURA 5: La pagina iniziale dell'applicazione FACILE.

necessità di usare i font prescritti dal manuale della *Corporate Identity* del Comune di Napoli, una soluzione basata sulle possibilità offerte da XeLaTeX, opportunamente "occultato" offrendo all'utente un *front-end* semplificato, è apparsa dunque la via ideale da percorrere.

Un evidente vantaggio derivante da questa impostazione è quello di poter avere uno strumento di lavoro condiviso, basato sull'esistenza di servizi informatici e archivi centralizzati a disposizione della platea di utenti, puntando a eliminare le differenze tra le prassi lavorative riscontrate nei diversi uffici.

3.1 Caratteristiche principali

La filosofia di sviluppo alla base di FACILE parte dalla considerazione che è necessario gestire migliaia di utenti che utilizzano sia sistemi operativi Windows che, in misura significativa, Linux (grazie alla migrazione voluta dal Comune di Napoli negli ultimi anni). L'esigenza che deriva da queste considerazioni ha spinto a sviluppare un sistema *server based*. Questo approccio semplifica drasticamente la gestione dei modelli da adottare e il flusso delle comunicazioni da produrre, attraverso la gestione centralizzata degli accessi, che vengono legati alla Intranet dell'Ente, e delle politiche di redazione delle comunicazioni (tipo di modello utilizzato dal singolo dipendente, flusso autorizzativo, aggiornamento dei modelli).

Si è scelto di sviluppare FACILE utilizzando il *Content Management System* (CMS) *open source* Drupal[2]. Questo *framework* di sviluppo ha permesso di implementare in tempi ridotti il *front-end* dell'applicazione e di gestire agevolmente l'inserimento dei contenuti delle comunicazioni, dei destinatari (da scegliere a partire da liste preimpostate e memorizzate in un *database* comunale) e dei parametri necessari al completamento della redazione e della spedizione dei documenti.

Si può immaginare FACILE come un'applicazione costituita da tre 'strati' software. Lo strato più esterno è l'interfaccia grafica, che è percepita dall'utente come una pagina web. Uno strato software molto importante è quello intermedio che raccoglie tutti i contenuti inseriti attraverso l'interfaccia grafica e li mette in relazione con le funzionalità di composizione del documento finale. Questo modulo del sistema non è altro che un *parser*, sviluppato in linguaggio PHP. Il suo scopo è di elaborare i contenuti inseriti dall'utente. Il risultato dell'elaborazione viene copiato al posto di alcune stringhe chiave contenute in un *template* di sorgente predisposto per essere compilato con XeLaTeX. Quest'ultimo rappresenta il cuore del terzo strato software, quello più interno.

Già in fase di progettazione e successivamente nel corso dello sviluppo dell'interfaccia grafica e del *parser* è stato utilizzato un approccio teso alla generalizzazione dei casi d'uso, prevedendo estensioni future dell'applicazione. A questo scopo sono stati definiti vari parametri di configurazione che rendono l'applicazione flessibile e consentono l'aggiunta di funzionalità supplementari senza modificarne la logica. Ad esempio, in un prossimo futuro i documenti generati con FACILE potranno essere firmati digitalmente, protocollati, inviati automaticamente ai destinatari via posta elettronica e archiviati nel sistema documentale del Comune di Napoli.

Il primo rilascio di FACILE risale all'aprile 2011. L'applicazione è stata installata su uno dei server dell'Intranet del Comune di Napoli insieme con la distribuzione TeX Live 2010.

2. `drupal.org`.

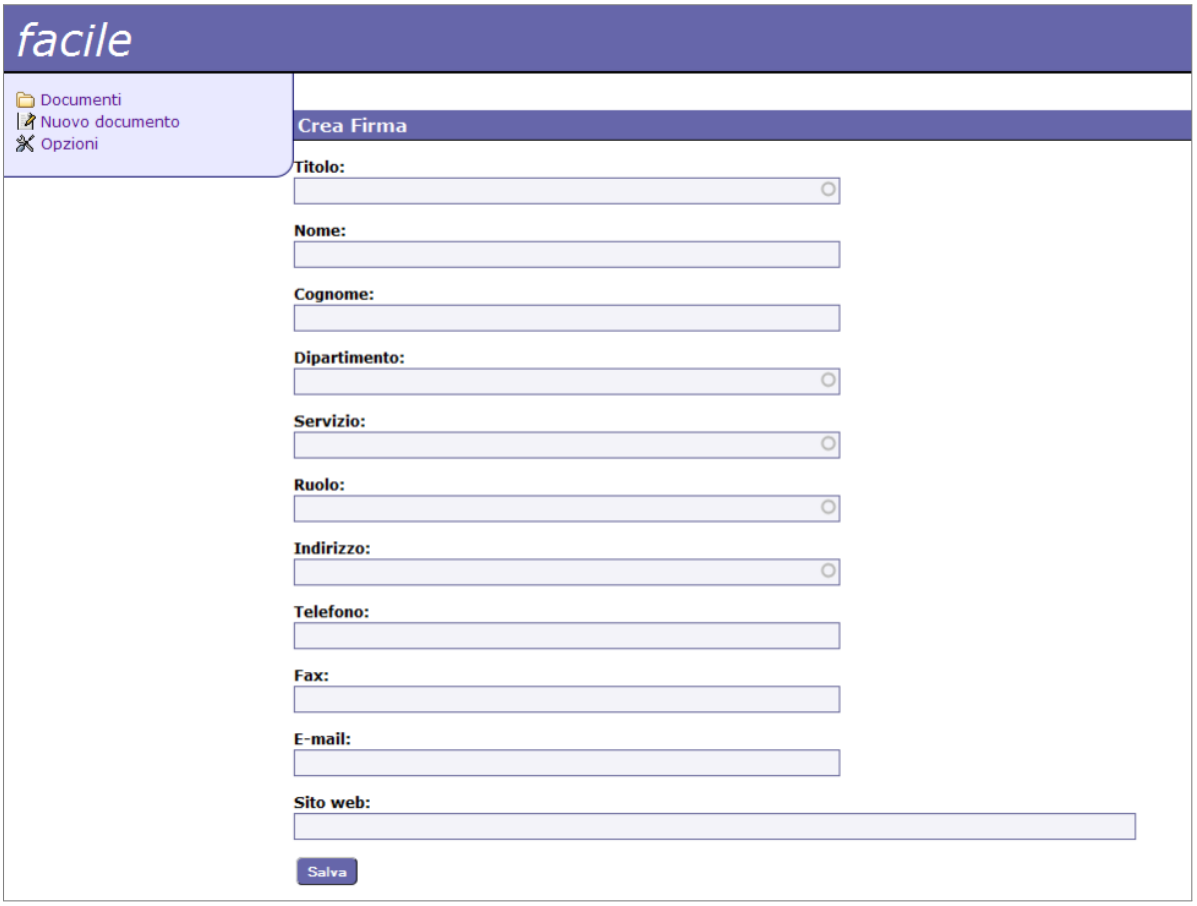


FIGURA 6: La pagina di configurazione della firma.

FIGURA 7: Schermata con la quale l'utente sceglie di creare un nuovo documento.

3.2 Breve panoramica sulle funzionalità

Per accedere a FACILE l'utente utilizza un *browser*, collegandosi alla pagina principale della Intranet del Comune di Napoli. Qui è disponibile una *tab* dedicata all'applicazione, come si vede dalla figura 4. Cliccando sulla *tab* l'utente, qualora autorizzato, accederà direttamente all'applicazione.

La pagina di accesso a FACILE è mostrata nella figura 5. La schermata principale di FACILE è composta da un menu sul lato sinistro della finestra e da un'area di lavoro costituita dalla parte rimanente della schermata. Il menu permette all'utente di creare un nuovo documento (cliccando sulla voce "Nuovo documento") e di gestire le proprie opzioni (voce "Opzioni"). Se l'utente ha già creato documenti, sul lato destro della schermata principale ne sarà visualizzato l'elenco recante l'oggetto, il destinatario, la data e i "tag" (ovvero delle categorie per organizzare e gestire con semplicità i documenti).

Nella parte alta della schermata principale sono presenti alcune icone per la gestione rapida di azioni sui documenti (modifica, duplicazione, eliminazione, eccetera).

Dal menu principale attraverso la selezione della scelta "Opzioni" si passa a delle schermate di configurazione del profilo utente. Un esempio è la schermata mostrata nella figura 6 con la quale è possibile definire una *firma*. La firma è importante poiché i dati ad essa associati vengono utilizzati, unitamente al logo dell'Ente, nella testatina e nel piede delle lettere. Ciascun utente può definire più di una firma — caratteristica utile quando la lettera viene materialmente preparata da una persona diversa da quella che compare ufficialmente come mittente. Ciascun utente ha la possibilità di memorizzare più di una firma all'interno del proprio profilo, scegliendo di volta in volta quella

appropriata alla lettera da mandare in stampa.

Cliccando sulla voce "Nuovo documento" del menu principale si accede ad una schermata come quella della figura 7. Questo passaggio consentirà in futuro di scegliere tra diversi tipi di documento, a seconda se la comunicazione è ad uso interno o esterno. Attualmente con FACILE è possibile comporre unicamente lettere di comunicazione ufficiali ad uso interno.

Cliccando sul pulsante "Crea" nella schermata della figura 7 si accede ad una schermata come quella visualizzata nella figura 8. All'utente viene presentato un *form* con un numero di campi da riempire. La struttura di questa finestra — una delle più importanti — rispecchia quella di una lettera. Sono presenti tutti gli elementi di cui ci si preoccupa quando si prepara una comunicazione ufficiale da diramare eventualmente a più destinatari.

Esaminando i diversi elementi della finestra ci si rende conto che si tratta di una sorta di *editor* con funzionalità particolari. Ad esempio, il primo campo è denominato "Intestazione A:". Esso corrisponde all'etichetta che si vuole utilizzare nella lettera per indicare il destinatario (solitamente l'etichetta consiste in "A:", ma potrebbe essere anche "Ai dirigenti:"). il campo successivo denominato "A:", invece, identifica il destinatario vero e proprio (ad esempio: "Tizio Caio, Via Tal dei Tali 123, 80100 Napoli"). Per inserire un destinatario si può procedere sia manualmente, riempiendo la casella di testo corrispondente, oppure, cliccando sul segno "+" a destra del campo "A:". In questo caso si ha accesso ad una rubrica residente sul server dell'applicazione, un *database* contenente le diciture ufficiali di tutti i possibili destinatari con i loro indirizzi. La rubrica può essere scorsa e l'utente può selezionare il destinatario cercato. I destinatari di questa sezione possono essere molteplici.

Un discorso analogo al precedente vale per il campo "CC:" riferito ai destinatari a cui inviare una copia in conoscenza del documento. Il campo "Instestazione CC:" corrisponde all'etichetta che si vuole utilizzare sul documento per indicare i destinatari secondari (solitamente l'etichetta consiste in "E, p. c.:"). Il campo "CC:", invece, identifica i destinatari secondari. Per inserire un destinatario a cui inviare una copia in conoscenza del documento si può procedere sia manualmente, riempendo la casella di testo corrispondente, oppure, cliccando sul segno "+" a destra del campo "CC:", scorrendo la rubrica dei contatti e selezionando il destinatario.

Dalle descrizioni precedenti è intuitivo comprendere il significato dei campi successivi: "Luogo", "Data", "Oggetto" e così via.

La sezione centrale del *webform* è il campo denominato "Corpo". Il corpo del documento è un vero e proprio *editor* che consente di inserire il testo della lettera e di formattarlo con gli appositi pulsanti. Il componente software che espone questa funzionalità[3] traduce il materiale immesso dall'utente in linguaggio HTML. Il *parser* di FACILE — lo strato software intermedio — raccoglierà questo codice traducendolo in codice LaTeX.

È quest'ultima una funzionalità sviluppata appositamente con l'applicazione. Attualmente tutti i principali costrutti di *markup* del linguaggio HTML sono tradotti nel *markup* del formato LaTeX (con possibilità di inserire immagini e di comporre correttamente anche delle tabelle non troppo complesse).

Una volta inserito il testo del documento, si può inserire una formula di chiusura, dopodiché è necessario selezionare la propria firma dall'apposito menu a tendina. I dati della firma vengono utilizzati nella intestazione della lettera, nell'area della firma e nel piedino del documento stesso.

Qualora si volessero aggiungere degli allegati al documento, è possibile cliccare sulla voce "File allegati", quindi selezionare uno o più file. Questa sezione permette di far comparire una lista eventuale di allegati in calce alla firma. L'utente ha la possibilità di inserire il testo che descrive i singoli allegati (che andrà effettivamente composto) e al tempo stesso di caricare i file corrispondenti. Questi ultimi saranno archiviati insieme a tutti i dati associati al documento.

Cliccando sul pulsante "Salva" sarà possibile salvare il lavoro svolto fino a quel momento e continuare l'inserimento dei dati del documento corrente senza creare il relativo file PDF. Per visualizzare l'anteprima del documento sarà possibile cliccare su "Anteprima". Altrimenti, per completare la procedura di creazione del documento, basterà cliccare sul pulsante "Salva e completa". In questo modo verrà creato il file PDF del documento corrente, scaricabile sul computer dell'utente e stampabile.

Quando l'utente termina il suo lavoro con la schermata della figura 8 il contenuto dei campi viene raccolto ed elaborato dal *parser*. Quando viene premuto il pulsante "Salva e completa" (oppure "Anteprima") il *parser* traduce l'input dell'utente e al tempo stesso accede ad un file `template.tex`. Quest'ultimo è memorizzato sul server dell'applicazione ed è unico per tutti gli utenti. Il *template* viene usato come base per la generazione di un file sorgente ⟨`nome-lettera`⟩`.tex`. Il nome unico del file sorgente viene costruito in base all'oggetto della lettera e alla data di creazione. Questo file non è accessibile agli utenti normali ma solo agli utenti con privilegio di amministratore. Ai primi sarà visibile solo il risultato della compilazione del sorgente con il programma `xelatex`, cioè il file ⟨`nome-lettera`⟩`.pdf`. Agli utenti amministratori sarà concesso sia di caricare eventuali aggiornamenti del file `template.tex` che accedere ai sorgenti delle singole lettere.

3. Si veda: `http://ckeditor.com`.

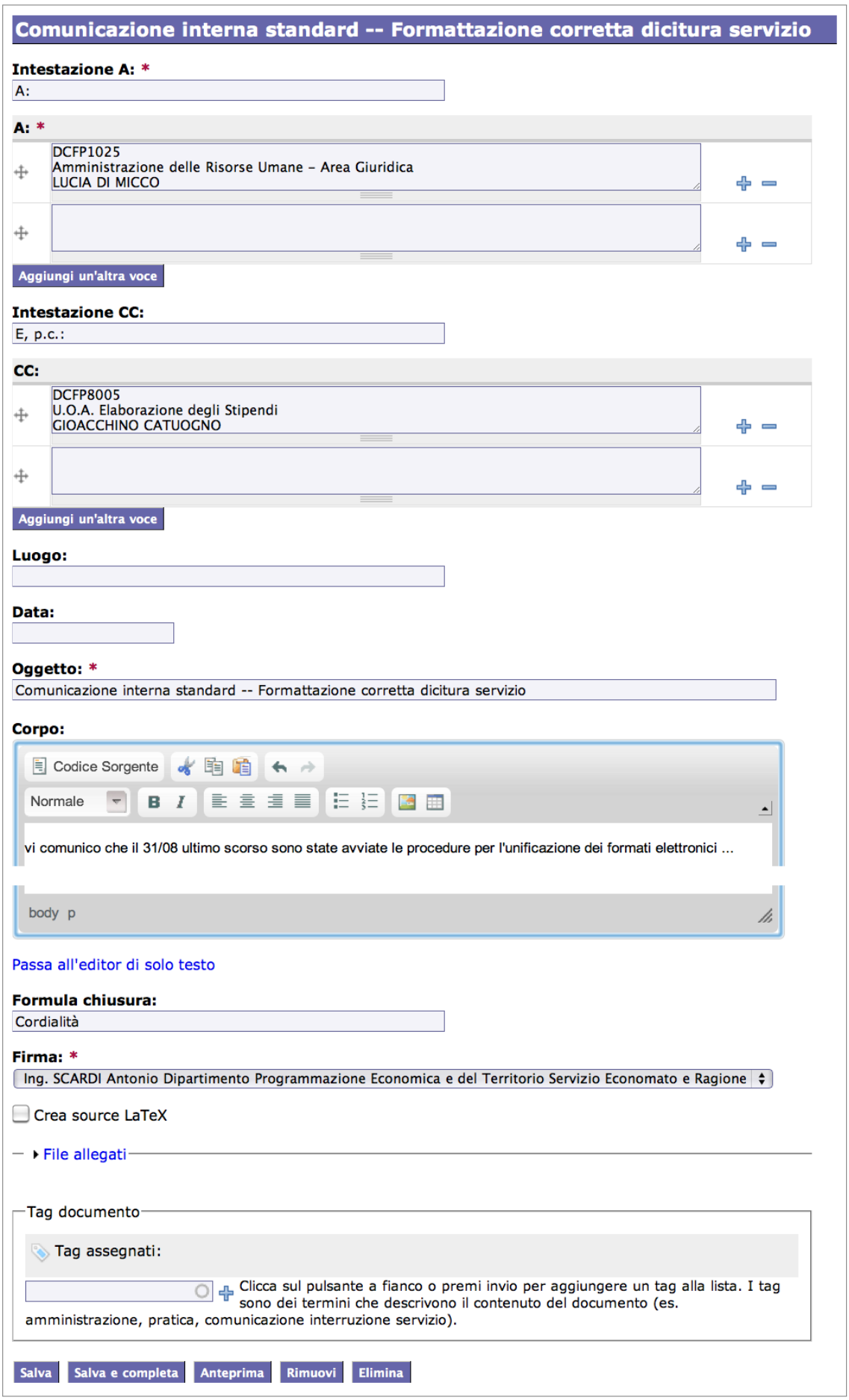


Figura 8: Creazione di un documento.

COMUNE DI NAPOLI

Dipartimento Comunicazione Istituzionale, Tecnologie e Società dell'informazione
Servizio *SIAD – Sistema Informativo, Amministrativo e Documentale*

Prot.

A: **Direzione Centrale I – Risorse strategiche e programmazione economico-finanziaria**
Servizio SIF – Sistema Informativo Finanziario
dott. Umberto MANDARA

Servizio RTI e Microinformatica
SEDE
dott.ssa Rosanna PERSICO

Data
31/08/2011

E, p.c.: **Direzione Generale**
Servizio Controllo qualità
dott. Massimo PACIFICO

Servizio Portale Web e nuovi media
SEDE
ing. Giuseppe CONTINO

Oggetto: **Attività di sperimentazione della web application FACILE**

Carissimi,

In merito alle attività in oggetto, che avranno inizio il giorno 23 settembre 2011, vi prego di dare massima diffusione possibile degli allegati alla presente.

Cordialità,

ing. Paolo Eugenio CRESCI
Responsabile Servizio Informatico

Allegati:

Allegato A. Regolamento e manuale d'uso della Intranet del Comune di Napoli. Aggiornamenti 2011.
Allegato B. Manuale utente dell'applicazione FACILE.

*Piazza Giovanni XXIII n. 6, 80126 Napoli, Italia · **tel.** (+39) 081 7958800 · **fax** (+39) 081 7958803*
sistema.informativo.amministrativo@comune.napoli.it** · **www.comune.napoli.it

FIGURA 9: Esempio di lettera per comunicazione interna prodotto con l'applicazione FACILE.

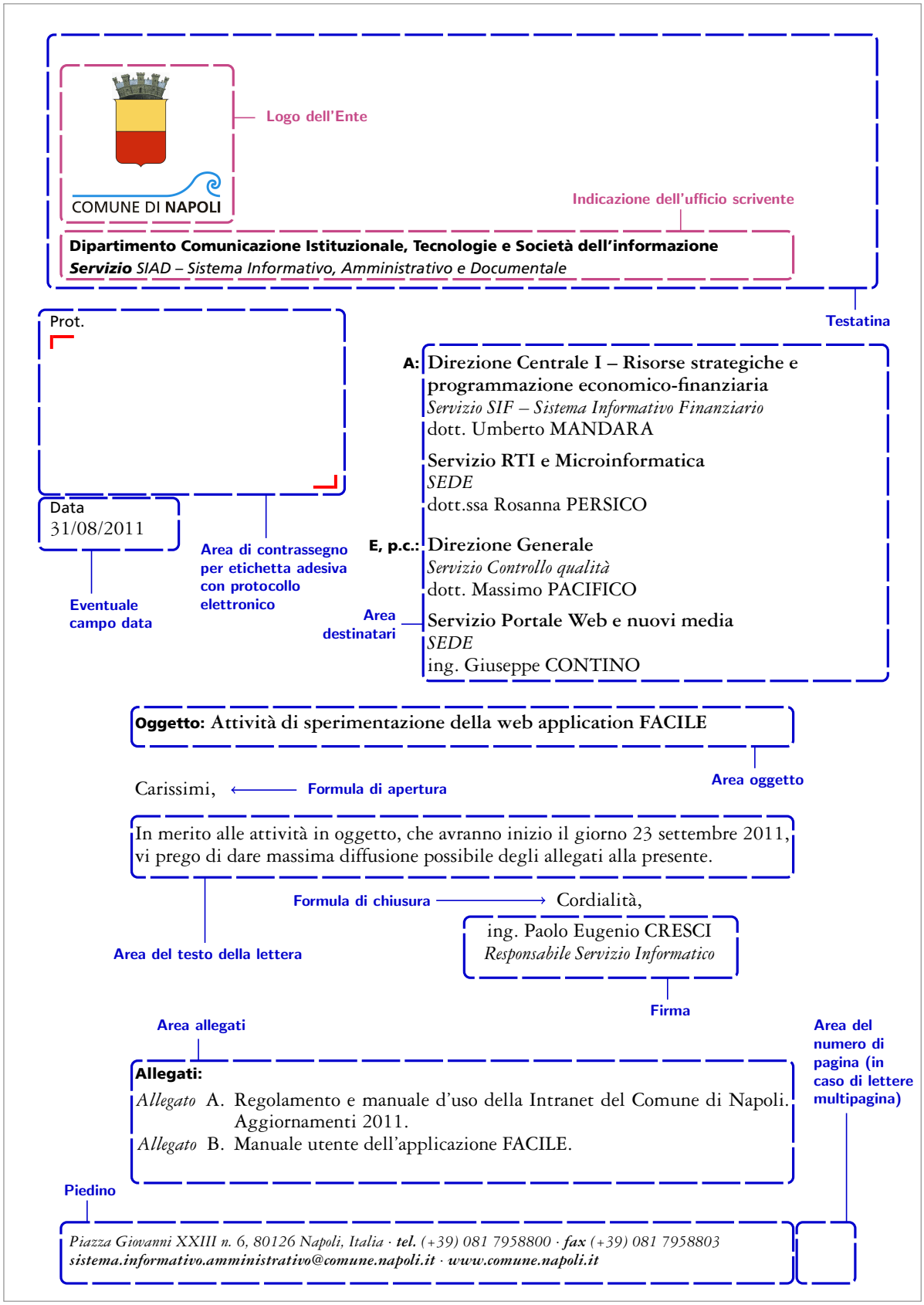
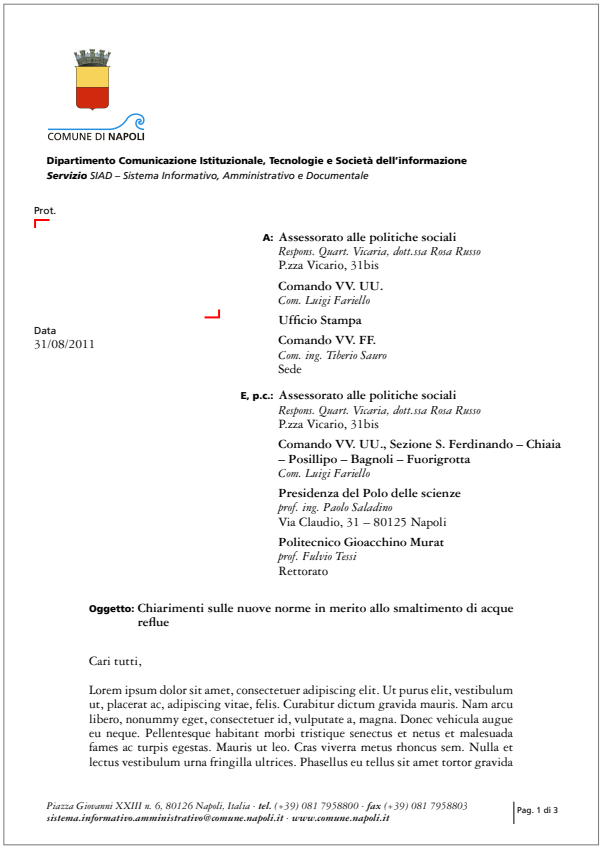


FIGURA 10: La medesima lettera mostrata nella figura 9. Sono evidenziati gli elementi che compongono la struttura tipografica del documento.



COMUNE DI NAPOLI

Dipartimento Comunicazione Istituzionale, Tecnologie e Società dell'informazione
Servizio *SIAD – Sistema Informativo, Amministrativo e Documentale*

Prot.

Data
31/08/2011

A: **Assessorato alle politiche sociali**
Respons. Quart. Vicaria, dott.ssa Rosa Russo
P.zza Vicario, 31bis

Comando VV. UU.
Com. Luigi Fariello

Ufficio Stampa

Comando VV. FF.
Com. ing. Tiberio Sauro
Sede

E, p.c.: **Assessorato alle politiche sociali**
Respons. Quart. Vicaria, dott.ssa Rosa Russo
P.zza Vicario, 31bis

Comando VV. UU., Sezione S. Ferdinando – Chiaia – Posillipo – Bagnoli – Fuorigrotta
Com. Luigi Fariello

Presidenza del Polo delle scienze
prof. ing. Paolo Saladino
Via Claudio, 31 – 80125 Napoli

Politecnico Gioacchino Murat
prof. Fulvio Tessi
Rettorato

Oggetto: **Chiarimenti sulle nuove norme in merito allo smaltimento di acque reflue**

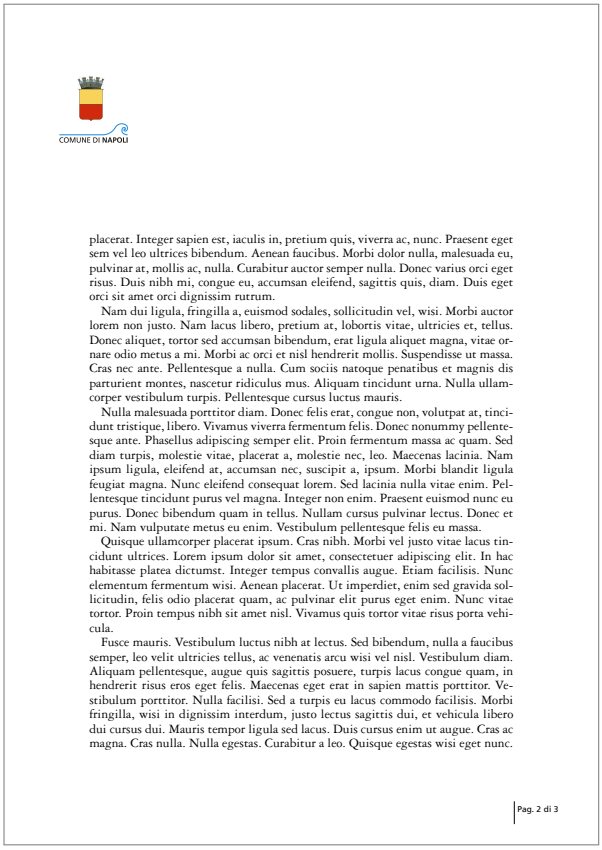
Cari tutti,

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida

*Piazza Giovanni XXIII n. 6, 80126 Napoli, Italia · **tel.** (+39) 081 7958800 · **fax** (+39) 081 7958803*
sistema.informativo.amministrativo@comune.napoli.it · www.comune.napoli.it

Pag. 1 di 3

(a) pagina 1/3



COMUNE DI NAPOLI

placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

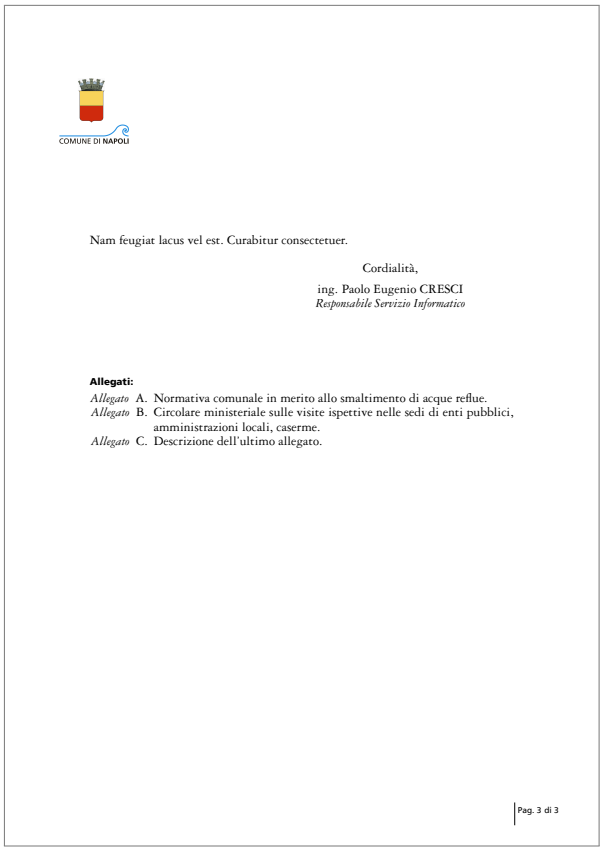
Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc.

Pag. 2 di 3

(b) pagina 2/3



COMUNE DI NAPOLI

Nam feugiat lacus vel est. Curabitur consectetuer.

Cordialità,

ing. Paolo Eugenio CRESCI
Responsabile Servizio Informatico

Allegati:
Allegato A. Normativa comunale in merito allo smaltimento di acque reflue.
Allegato B. Circolare ministeriale sulle visite ispettive nelle sedi di enti pubblici, amministrazioni locali, caserme.
Allegato C. Descrizione dell'ultimo allegato.

Pag. 3 di 3

(c) pagina 1/3

FIGURA 11: Esempio di lettera multipagina prodotta con l'applicazione FACILE.

Nella figura 9 è mostrato un esempio di lettera monopagina prodotto con FACILE. La figura 10 mostra la stessa lettera e mette in evidenza i diversi elementi tipografici. Se essa viene confrontata con la schermata di FACILE della figura 8 si comprende quanto sia importante che il *parser* dell'applicazione sia efficiente e che il *template* di sorgente LaTeX sia in grado di restituire risultati ben composti per un ventaglio potenzialmente molto ampio di casi d'uso.

La figura 11 è un esempio di lettera lunga più di una pagina. Nel piedino, a destra, è riportata la numerazione riferita al numero totale di pagine ("1 di 3", "2 di 3", "3 di 3"). La gestione di questa specifica, oltre al caricamento di un apposito pacchetto, richiede di gestire delle compilazioni multiple con XeLaTeX.

Il numero di compilazioni è un tipo di configurazione impostabile dagli utenti di FACILE con privilegio di amministratore dell'applicazione. Gli utenti amministratori hanno accesso ad un menu "Opzioni" più ricco di voci. Inoltre, un utente amministratore può anche scegliere di far restituire all'applicazione il sorgente LaTeX, un file `.tex` anziché il file in formato PDF. Ciò è utile quando l'amministratore vuole effettuare un *debug* del processo di composizione.

3.3 Il file *template* di una comunicazione interna

Il *template* di sorgente LaTeX che viene elaborato dal *parser* di FACILE è una parte fondamentale dell'applicazione. Nella sua attuale versione, esso consta di circa 1500 linee di testo, che comprende il codice vero e proprio e le linee di commento. Il commento del codice è stato inserito con lo scopo di fornire una guida rapida agli utenti esperti che vogliono effettuare operazioni di *debug*. I commenti facilitano la lettura del codice in generale ma, soprattutto, documentano le svariate macro presenti nel *template*, alcune delle quali scritte utilizzando comandi primitivi del linguaggio TeX.

Il sorgente è basato sulla classe scrlttr2 appartenente al *bundle* di pacchetti di estensione KOMA-Script (Kohm e Morawski, 2011). I pacchetti di KOMA-Script sono stati sviluppati dagli autori tedeschi Markus Kohm e Jens-Uwe Morawski e sono stati scelti in base al loro alto livello di configurabilità, alla robustezza del codice e in base alla ampia comunità di utenti sparsi in tutto il mondo. Questo è un aspetto importante perché assicura un'ampia base di conoscenza con buone possibilità di trovare soluzioni a problemi specifici[4].

3.3.1 Una implementazione basata sulla classe scrlttr2

Tra le classi definite da KOMA-Script vi è la scrlttr2, dedicata alla composizione di lettere sia private che commerciali.

Rilasciata nel 2002, scrlttr2 è una classe che reimplementa *ex novo* le funzionalità della classe standard letter di LaTeX. Anche se parte del codice di scrlttr2 è identico a quello delle principali classi di KOMA-Script, le lettere sono molto diverse dagli articoli, dai saggi o dai libri. Ciò giustifica la presenza nel *bundle* KOMA-Script di una classe espressamente dedicata alle lettere. Inoltre, essendo stata progettata a partire da zero, scrlttr2 fornisce un'interfaccia d'uso alquanto particolare rispetto alle normali classi scritte per LaTeX; interfaccia che conferisce alla classe una grande flessibilità.

Una caratteristica particolare della classe scrlttr2 è data dalla possibilità di cambiare molte opzioni, anche dopo il caricamento con `\documentclass`. Il comando `\KOMAoptions` serve a tal fine, accettando una lista di opzioni assegnate secondo lo schema $\langle \texttt{chiave} \rangle = \langle \texttt{valore} \rangle$ (grazie al pacchetto keyval).

Si rimanda il lettore al manuale d'uso della classe per approfondimenti sulle sue opzioni di personalizzazione.

Va osservato che il sorgente che deve scaturire dal *template* deve poter essere usato per comporre lettere di una o di più pagine. Inoltre non sono rari i casi in cui le comunicazioni vanno diramate a più di un destinatario principale ed in copia per conoscenza a numerosi destinatari secondari. Può capitare in alcuni casi che la lista completa dei destinatari preveda anche più di un salto di pagina.

Grazie a delle macro appositamente scritte e a delle particolari impostazioni della classe scrlttr2, il codice del *template* implementa un modello generale di lista dei destinatari con eventuali salti pagina. Per una lista di destinatari di dimensioni maggiori dello spazio disponibile sulla prima pagina il contenuto della lista viene spezzato in più parti; la prima parte viene composta nella prima pagina, la seconda parte sulla seconda pagina ed eventualmente le parti rimenenti sulle pagine successive.

Dopo la lista dei destinatari verranno collocati l'oggetto della comunicazione e la frase di apertura, — ad esempio "Cari tutti," — seguiti dal corpo del documento, dalla frase di chiusura — ad esempio "Cordialità" —, dalla firma e dall'eventuale lista di allegati.

Si deve osservare cha nella prima fase di sperimentazione si è preferito che la sigla dello scrivente dovesse essere fisicamente apposta a penna una volta stampata la lettera. In futuro l'inserimento di un'immagine con una firma digitalizzata è semplice questione di uso del comando `\includegraphics`.

Il logo del Comune di Napoli è posizionato nello spazio della testatina (*heading*). La prima testatina è diversa da quelle che compaiono sulle pagine successive alla prima. Questa specifica del manuale della *corporate identity* è direttamente configurabi-

4. Si veda: `http://www.komascript.de`.

le attraverso l'interfaccia della classe scrlttr2, che prevede un'apposita impostazione per le testatine successive alla prima. In ogni caso le testatine sono composte inserendo il nome del 'Dipartimento' e del 'Servizio' scrivente. Queste informazioni sono impostate ad alto livello dall'utente di FACILE attraverso la schermata di configurazione della firma (figura 6). Analoghe informazioni vengono inserite nel piedino.

Sulla prima pagina è mostrato un riquadro per l'apposizione dello *sticker* che reca il codice del protocollo e la data (necessario per tutte le comunicazioni ufficiali destinate all'archiviazione nel sistema documentale dell'Ente). Al momento questa operazione avviene in seguito alla stampa del documento, cioè l'etichetta adesiva con il protocollo non è generata contestualmente alla lettera ma proviene fisicamente da un apposito ufficio. Una estensione futura di questa funzionalità consiste nel reperimento *online* del codice del protocollo, direttamente in fase di finalizzazione del documento in modo che la composizione di questo elemento (codice a barre incluso) avvenga insieme a tutti gli altri attraverso XeLaTeX.

Lo spazio per la firma è lasciato in bianco. Sotto lo spazio per la sigla (che per il momento deve essere apposta a penna sul documento stampato) compare il nome del firmatario e la sua qualifica (ruolo). Una estensione futura prevede l'inserimento di una firma digitale, configurabile ad alto livello attraverso il *front-end*, inserita a basso livello dal motore di composizione.

Data la dimensione del file *template*, di seguito si riporteranno solo alcune tra le soluzioni implementative adottate nel codice.

3.3.2 Qualche dettaglio sul codice sorgente

La parte iniziale del template ha un preambolo strutturato come segue (con evidenziazione separata per i comandi introdotti dalla classe scrlttr2):

```
\documentclass[paper=a4,fontsize=12pt,version=last
    ]{scrlttr2}

% impostazioni aggiuntive della classe scrttr2
\KOMAoptions{enlargefirstpage,twoside=false,
     foldmarks=false,fromalign=left,fromrule=false,
      pagenumber=footright}

% Pacchetti specifici per XeLaTeX
\usepackage{fontspec,xltxtra,xunicode}
\defaultfontfeatures{Scale=MatchLowercase,
   Mapping=tex-text}
\setmainfont[Mapping=tex-text]{
   Garamond 3 LT Std}
\setsansfont[Mapping=tex-text]{
   Frutiger LT 55 Roman}
\usepackage{polyglossia}
\setdefaultlanguage{italian}

% Pacchetti aggiuntivi
\usepackage{calc,xifthen,xspace,url}
\urlstyle{same}
\usepackage{graphicx,xcolor,varwidth,relsize,
     ragged2e,paralist,marvosym,eurosym,textcomp,
     alphalph,pagesLTS}

% Alcuni termini italiani (vedi "scrguien.pdf" p. 212)
\providecaptionname{italian}\phoneofficename{
     ufficio}
\providecaptionname{italian}\phonelaboratoryname{
     lab.}
\providecaptionname{italian}\phonemobilename{
     cellulare}
\providecaptionname{italian}\phoneprivatename{
     privato}
\providecaptionname{italian}\phoneemergencyname{
     emergenza}

% Impostazioni generali riguardanti il layout (vedi "scrguien.
     pdf" p. 185)
\pagestyle{headings}
% ...
```

Seguono delle impostazioni riguardanti l'aspetto delle testatine e dei piedini:

```
\makeatletter
% larghezza della prima testatina
\@setplength{firstheadwidth}{\paperwidth}
\@addtoplength{firstheadwidth}{-30mm}
% larghezza del primo piedino
\@setplength{firstfootwidth}{\useplength{
     firstheadwidth}}
% posizione della prima testatina
\@setplength{firstheadhpos}{15mm}
\@setplength{firstheadvpos}{15mm}
\addtolength{\textheight}{5mm}
% posizione del primo piedino
\@addtoplength{firstfootvpos}{1mm}
\@setplength{firstfoothpos}{\useplength{
     firstheadhpos}}
% posizione orizzontale della lista
\@setplength{toaddrhpos}{15mm}
% posizione verticale della lista
\@setplength{toaddrvpos}{80mm}
% larghezza della scatola contenente
\@setplength{toaddrwidth}{28.5cm}
% indentazione della lista dei destinatari
\@setplength{toaddrindent}{8.2cm}
% indirizzo di risposta
\@setplength{backaddrheight}{0pt}
% larghezza box informazioni supplementari
\@setplength{locwidth}{5cm}
% posizione orizzontale box informazioni supplementari
\@setplength{lochpos}{%
   21cm-\useplength{locwidth}-\useplength{
     firstheadhpos}}
% posizione verticale box informazioni supplementari
\@setplength{locvpos}{52mm}
% offset orizzontale dell'area−firma
\@setplength{sigindent}{8cm}
% offset verticale dopo la frase di chiusura
\@setplength{sigbeforevskip}{6pt}
\makeatother
```

Successivamente si hanno i comandi che impostano l'area di stampa in alto a sinistra della prima pagina, subito al di sotto dell'intestazione con il logo e il Dipartimento scrivente (si veda la figura 10). Nel codice seguente si osserva il modo in cui si effettua il disegno del contrassegno rappresentante l'area destinata allo *sticker* con il protocollo. Suc-

cessivamente si riporta anche il codice che gestisce l'eventuale campo non nullo contenente la data:

```
% Area−Riferimenti: protocollo, data, ecc.
\removereffields
% variabile date resa nulla in modo da definirne una locale (
    vedi "myref1") per consentire la personalizzazione dell'
    area−riferimenti
\setkomavar{date}{\null}
\providecaptionname{italian}\datename{}

% MYREF
% (vedi refwidth, refvpos, refhpos in "scrguien.pdf" p. 201)
% Area−protocollo
\providecaptionname{italian}\myrefname{\relsize{2}
    Prot.}
\setkomavar{myref}{%
  \makebox[6.7cm][l]{% vedi refwidth
    % riquadro per l'apposizione dello sticker
    % del protocollo (6.7 x 3.2 cm)
    \mbox{}%
    \color{red}%
    \raisebox{3.2cm}{%
      \rule{2pt}{9pt}\rule[7.0pt]{1em}{2pt}%
    }%
    \mbox{}\hfill\mbox{}%
    \makebox[0pt][r]{%
        \rule{0pt}{3.2cm}\rule[0.0pt]{1em}{2pt
    }\rule{2pt}{9pt}
}}}
\addtoreffields{myref}

% MYREF1
% per un campo data non vuoto viene stampata la
% stringa di etichettatura ''Data''
% per un campo data vuoto la stringa diventa nulla
\newboolean{FCLdateIsEmpty}
\setboolean{FCLdateIsEmpty}{true}
\def\tempDate#1{%
   \ifthenelse{\isempty{#1}}%
       {}%
       {\relsize{2}Data%
        \setboolean{FCLdateIsEmpty}{false}}
}%
\def\tempDatea{%
  \tempDate{%
  %$field_date%
  }%
}
\newkomavar[\tempDatea]{myref1}
\setkomavar{myref1}{%
  \makebox[4cm][l]{% see refwidth
      %$field_date%
  }%
}
% aggiunge effettivamente la data all'area−riferimenti
\addtoreffields{myref1}
```

Il template non è un sorgente direttamente compilabile. Esso viene utilizzato per produrre un sorgente compilabile dopo che alcune stringhe segnaposto sono state opportunamente sostituite. Le stringhe segnaposto sono state definite secondo il seguente schema:

%$field_⟨*nome-campo*⟩

La parte ⟨*nome-campo*⟩ corrisponde ad uno dei campi della schermata di FACILE mostrata nella figura 8. Il parser cattura il contenuto testuale del campo e lo sostituisce alla stringa segnaposto corrispondente, presente nel template. Si consideri, ad esempio, il campo della data. Questo è anche un caso particolare di gestione avanzata della sostituzione. La webform propone all'utente un campo data che non sempre è necessario riempire, perché la data è presente sullo *sticker* del protocollo che verrà applicato una volta stampata la lettera. Quindi è necessario un algoritmo che controlli se la stringa della data catturata dal parser è vuota. In caso positivo lo spazio della lettera riservato alla data rimane completamente bianco. Nel caso in cui l'utente immette una data il campo **%$field_date** potrà essere sostituito con la stringa corrispondente — nell'esempio della figura 9 la stringa è "31 agosto 2011". Nel codice precedente questo algoritmo è implementato nelle macro `\tempDate` e `\tempDatea`. Nell'esempio citato l'utente ha impostato una data non vuota e in fase di composizione viene stampata la stringa "Data" in una posizione opportuna del foglio e sotto di essa la stringa "31 agosto 2011" (si veda la figura 10).

Proseguendo a esaminare il preambolo del file *template*, nel codice seguente si imposta il posizionamento di alcuni elementi aggiuntivi:

```
% posizione e dimensioni dell'area riferimenti
\makeatletter
%
\@setplength{refhpos}{%
   0.7\useplength{firstheadhpos}
}
\@setplength{refvpos}{70mm}
\@setplength{refwidth}{50mm}
\@setplength{refaftervskip}{2cm}
%
\makeatother
```

Seguono altre impostazioni:

```
% Dipartimento
\newkomavar{DeptName}
\setkomavar{DeptName}{%
   %$field_department_name%
   \mbox{}%
}
% Servizio
\newkomavar{ServName}
\setkomavar{ServName}{%
   %$field_service_name%
   \mbox{}%
}

% Mittente, (va nell'area−firma)
\setkomavar{fromname}{%
   %$field_signature%
   \mbox{}
}

% Area−firma
\setkomavar{signature}{%
   %$field_signature%
   \mbox{}
   \\
   \textit{%
      %$field_role_signature%
   }%
}
```

```
% Testatina in prima pagina (First page header)
\makeatletter
\firsthead{%
   \begin{tabular}{@{}l@{}}
      \makebox[0.0\textwidth][l]{%
         \includegraphics[%
            width=3.5cm%
            ]{%
            LOGO_COMUNE_NAPOLI.png%
         }%
      }%
      \\
      \parbox{1.0\useplength{firstheadwidth}}{%
         \raggedright
         \sffamily\bfseries\normalsize%
         \usekomavar{DeptName}\\
         {%
            \relsize{0}%
            \usekomavar{ServName}%
         }%
      }%
   \end{tabular}
}
\@setplength{fromrulethickness}{0pt}
\makeatother

% PIEDINI e FIRMA, macro con gestione di campi vuoti
\newcommand{\FCLaddressSignature}[1]{%
  \ifthenelse{\isempty{#1}}%
    {}% if #1 is empty
    {#1}% if #1 is not empty
}
\newcommand{\FCLtelSignature}[1]{%
  \ifthenelse{\isempty{#1}}%
    {}% if #1 is empty
    {$\cdot$ {\bfseries tel.} #1}% if #1 is not empty
}
\newcommand{\FCLfaxSignature}[1]{%
  \ifthenelse{\isempty{#1}}%
    {}% if #1 is empty
    {$\cdot$ {\bfseries fax} #1}% if #1 is not empty
}
\newcommand{\FCLemailSignature}[1]{%
  \ifthenelse{\isempty{#1}}%
    {}% if #1 is empty
    {\bfseries\url{#1}}% if #1 is not empty
}
\newcommand{\FCLurlSignature}[1]{%
  \ifthenelse{\isempty{#1}}%
    {}% if #1 is empty
    {$\cdot$ {\bfseries\url{#1}}}% if #1 is not empty
}

% Piedino della prima pagina (First footer)
\makeatletter
\firstfoot{%
    \parbox{16.51cm}{%  magic number, larghezza box
     contenente footer a cui si affianca il numero di pagina
       \normalfont\footnotesize\itshape
          \FCLaddressSignature{%
             %$field_address_signature%
          }\xspace
          \FCLtelSignature{%
              %$field_tel_signature%
          }\xspace
          \FCLfaxSignature{%
              %$field_fax_signature%
          }\xspace
          \\
          \FCLemailSignature{%
              %$field_email_signature%
          }\xspace
          \FCLurlSignature{%
              %$field_web_signature%
          }
    }% ... end of parbox
    % Numero di pagina
    \ifnum\value{pagesLTS.pagenr}>1%
    \makebox[0pt][l]{% magic numbers, aggiustamenti
     verticali
        \raisebox{3pt}[0pt][0pt]{%
            \rule[3pt-\f@baselineskip]%
               {0.8pt}{8pt+\f@baselineskip}%
            \rule{3pt}{0pt}% spacer
            \relsize{-1}\sffamily\upshape
            \pagemark di \pageref{LastPage}
        }% ... raisebox
    }% ...makebox
    \fi
}
\makeatother
```

Il codice precedente introduce delle macro come `\FCLaddressSignature`, `\FCLtelSignature`, `\FCLfaxSignature`, eccetera, che gestiscono dei campi lasciati eventualmente in bianco dall'utente nella definizione della firma. I campi nulli non vengono riportati nel piedino, la cui composizione procede senza errori in tutti i casi d'uso.

Infine, si riporta un estratto del resto del codice:

```
% gestione margini, ecc.
%
\usepackage[left=3cm,right=3cm,top=8cm,bottom=3cm]
     %
    {geometry}
\setlength{\footskip}{1.0cm}

\begin{document}
%
\pagenumbering{arabic}% richiesto dal pacchetto
     pagesLTS

\FCLcheckCcAddressListEmpty{%
    % assegna \boolean{FCLccAddressListIsEmpty}
    %$field_address_cc%
}%
% ...

\setkomavar{subject}{%
     % controlla se va spezzata la lista
     \ifx\FCLaddressListSplitted\undefined\relax
          \ifdim\EZTsizePopBoxa >4.6cm
               \rule{0pt}{\FCLsizePopBoxa-5.5cm}\
    par% riga distanziatrice verticale
               \mbox{}%
          \fi
     \else
          % lista spezzata
          \mbox{}%
          %
          \newpage%..... salto pagina
          % ...
          % ...
     \fi
     % ORA COMPONE L'OGGETTO
     \noindent
     \mbox{%
          {\sffamily Oggetto:}\ \parbox[t]{\
    textwidth-1.7cm\relax}{%
               \noindent
```

```
                \hyphenpenalty=5000
                \tolerance=1000
                %$field_object%
            }%
        }%
}

\begin{letter}{
% ...
% FORMULA DI APERTURA
\FCLcheckOpeningEmpty{%
    %$field_opening%
}

\opening{%
     %
     %$field_opening%
     \mbox{}\ifthenelse{\boolean{FCLopeningIsEmpty
     }}{}{,}%
}

% ------------------------------
     BEGIN LETTER BODY
%
\noindent
%
%$field_body%
%
% ------------------------------
     END LETTER BODY
%
%  CLOSING PHRASE
%
\FCLcheckClosingEmpty{%
    %$field_closing_phrase%
}%

\closing{%
   \enlargethispage*{2.0\baselineskip}% starred
     version tries to squeeze material in page
   \mbox{}
   %
   %$field_closing_phrase%
   %
   \ifthenelse{\boolean{FCLclosingIsEmpty}}{}{,}%
   \mbox{}
}
%
% ALLEGATI
\FCLcheckAttachmentsListEmpty{%
   %$field_attachments%
}
%
%
\ifthenelse{\boolean{FCLattachmentsListIsEmpty}}{}{
     %
%
% Allegati presenti
%
\noindent\mbox{}\rule{0pt}{3.0\baselineskip}%
     spacer
\\
\noindent{\sffamily\bfseries Allegati:}\\
\begin{compactenum}[\mbox{\itshape Allegato\ } A.]
     %
    \pprowsAttachments{}{%
        %$field_attachments
   }
\end{compactenum}
%
}% ...endif list is empty
%
\end{letter}
\end{document}
```

4 Conclusioni

È stata presentata la *web application* FACILE, un'applicazione *server-based* che utilizza la distribuzione TEX Live per comporre comunicazioni ufficiali ad uso interno dell'Amministrazione comunale di Napoli. Le lettere prodotte con questo strumento sono conformi alle specifiche del Manuale della *Corporate Identity* del Comune di Napoli. Si tratta probabilmente del primo esperimento in Italia in cui si utilizza LATEX a questo livello in una pubblica amministrazione.

Ringraziamenti

Si ringrazia il Comune di Napoli, la Engineering Ingegneria Informatica S. p. a. e la Kelyon S. r. l. per il supporto tecnico offerto durante la stesura di questo articolo. Si ringrazia inoltre Enrico Gregorio per i suggerimenti sull'implementazione di alcune macro, forniti attraverso il forum di GUIT.

Riferimenti bibliografici

Anonimo (2006). «Manuale della Corporate Identity del Comune di Napoli». Comune di Napoli, Dipartimento Comunicazione Istituzionale e Immagine, http://www.comune.napoli.it.

Kohm, M. e Morawski, J.-U. (2011). «KOMA-script a versatile LATEX 2ε bundle». Disponibile su www.ctan.org.

▷ Agostino De Marco
Università degli Studi di Napoli "Federico II"
Dipartimento di Ingegneria Aerospaziale
Via Claudio 21
80125 Napoli
agostino dot demarco at unina dot it

▷ Paolo Eugenio Cresci
Dirigente Informatico
del Comune di Napoli
Servizio SIAD – Sistema Informativo Amministrativo e Documentale
Piazza Giovanni XXIII 6
80126 Napoli
paoloeugenio dot cresci at comune dot napoli dot it

Presentazioni da ArsTEXnica numero 11

Il numero di lavori presentati per il GUIT *meeting*2011 è stato notevolmente inferiore a quello delle scorse edizioni. Abbiamo ritenuto opportuno chiedere agli autori che hanno pubblicato un articolo su ArsTEXnica numero 11 di presentare comunque il loro lavoro al meeting.

Riportiamo qui di seguito gli abstract degli articoli, leggibili questi ultimi integralmente su ArsTEXnica numero 11, i cui autori hanno cortesemente accettato la nostra proposta.

1 Introduzione agli strumenti per grecisti classici

Gianluca Pignalberi, Salvatore Schirone, Jerónimo Leal

Il lavoro del grecista classico è parimenti difficile, o anche di più, di quello di un qualunque altro letterato. Per questo ha bisogno di strumenti di scrittura piuttosto sofisticati e potenti. Questo articolo presenta due editor, due compositori e alcune tecniche immediatamente utili per quel tipo di lavoro, e tutti utilizzati proficuamente dagli autori del presente articolo.

2 Extending ConTEXt-mkiv with PARI/GP

Luigi Scarso

Questo articolo mostra come costruire un *binding* a PARI/GP, un noto sistema di computer algebra, per ConTEXt-mkiv; mostra inoltre alcuni esempi di come risolvere alcuni problemi algebrici basilari.

An Introductive Presentation of xsl-fo

Jean-Michel Hufflen

Sommario

This short statement aims to sketch the broad outlines of the presentation performed at the GUIT 2011 meeting.

Keywords XML, XSL-FO, XSLT, fop.

Sommario

Questa breve relazione si propone di dare un'esposizione a grandi linee dell'intervento tenuto al GUIT meeting 2011.

Parole chiave XML, XSL-FO, XSLT, fop.

1 Introduction

XSL-FO[1] is an XML[2] dialect, aiming to describe high-quality output prints. The current 'official' W3C[3] recommendation is (W3C, 2006), describing XSL-FO 1.1, although many introductory books—such as (Pawson, 2002)—are based on the first version (1.0) (W3C, 2001), but changes are slight. There are also some design notes for a future version (2.0) (W3C, 2011), but at the time of writing, they have not been finalised yet as a recommendation.

Roughly speaking, XSL-FO shares some features with (LA)TEX: in particular, XSL-FO processors are not WYSIWYG[4], like Microsoft Word. The tasks performed by such an XSL-FO processor may be viewed as compiling a source text, as in (LA)TEX. However, there are also big differences. First, the markup is more systematic within XSL-FO, since it is an XML dialect. Second, XSL-FO is more verbose than (LA)TEX, it has not been designed for direct use. The best way to take advantage of XSL-FO features is to write XSLT[5] stylesheet (W3C, 1999, 2007) that result in XSL-FO source text. That is, this *modus operandi* aims to separate data and layout. Last but not least, XSL-FO has been designed to be able to deal with any written text, using any language. So XSL-FO source texts can use Unicode characters unicode2006b, and in particular, an XSL-FO processor should provide an implementation of the Unicode bidirectional algorithm (Unicode Consortium, 2008).

1. e**X**tensible **L**anguage **S**tylesheet—**F**ormatting **O**bjects.

2. e**X**tensible **M**arkup **L**anguage. For several years, XML has become a central formalism for data interchange. Readers interested in an introductory book to this formalism can consult (Ray, 2001).

3. **W**orld **W**ide **W**eb **C**onsortium.

4. **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et.

5. e**X**tensible **S**tylesehet **L**anguage **T**ransformations.

In (Hufflen, 2011), we show that getting started with XSL-FO is quite easy for users experienced with LATEX. This article is the Italian translation of (Hufflen, 2008), a revised and expanded version of (Hufflen, 2007). Some technical points related to multidirectional typesetting in XSL-FO are described in (Hufflen, 2009).

2 Plan

Our presentation will use Apache fop[6] (apachefop 2011) as XSL-FO processor. Its last version (1.0) is XSL-FO 1.1-compliant. In fact, no XSL-FO processor implements the whole of the W3C recommendation (W3C, 2006), but FOP, written in Java (java), is free software, easy to use, and implements most of features related to Western-European languages. It can run an XSLT stylesheet generating XSL-FO texts and typeset the result, but only for XSLT 1.0 stylesheets[7] (W3C, 1999). Let us notice that knowing XSLT—preferably XSLT 2.0, more powerful—is very useful if you would like to use XSL-FO, because some tasks can be only performed at this step. For example, XSL-FO can specify footnotes, but does not provide ways to number them automatically. Likewise, it can typeset indexes, but you are in charge of filling them in. The same for a table of contents. However, these tasks may be automatised, but at the level of an XSLT processor applied to an XML document. More generally, XSL-FO can specify how a text has to be typeet, but cannot *add* any information to an existing text.

In a first part, our presentation will focus on the `fo:block` element of XSL-FO, used to specify paragraphs, and comparable with LATEX's `minipage`. We will show how some usual constructs in LATEX can be put into action by means of nested elements `fo:block` and `fo:inline` of XSL-FO. Then we will go on with an introduction to formatting *lists* and *tables*.

In a second part, we will go thoroughly into the notion of *page models*, comparable to document classes in LATEX 2_ε. Roughly speaking, document classes (.cls files) are just complete styles among which a LATEX user chooses before writing a document, whereas page models are built by assembling simple elements with accurate attributes.

6. 2011 **F**ormatting **O**bjects **P**rocessor.

7. In practice, this is not a severe limitation: you can 'manually' apply an XSLT 2.0 stylesheet (W3C, 2007) by means of an accurate processor before calling FOP. Besides, no XSL-FO processor is directly associated with an XSLT 2.0 processor, at the time of writing.

We will end with an introduction to multidirectional typesetting, based on some examples already given in (Hufflen, 2009).

Acknowledgements

Thanks to Massimiliano Dominici for the abstract's Italian translation.

Riferimenti bibliografici

apachefop 2011. *Apache* FOP. `http://xmlgraphics.apache.org/fop/`, August 2011.

Jean-Michel Hufflen. Introducing LaTeX users to XSL-FO. TUG*boat*, 29(1):118–124, 2007. EuroBachoTeX 2007 proceedings.

Jean-Michel Hufflen. Passer de LaTeX à XSL-FO. *Cahiers* GUT*enberg*, 51:77–99, October 2008.

Jean-Michel Hufflen. Multidirectional typesetting in XSL-FO. In Tomasz Przechlewski, Karl Berry, and Jerzy B. Ludwichowski, editors, *TEX: at a Turning Point, or at the Crossroads? Proc. BachoTEX 2009 Conference*, pages 37–40, April 2009.

Jean-Michel Hufflen. Passare da LaTeX a XSL-FO. *ArsTEXnica*, 11:41–53, April 2011. Italian translation of (Hufflen, 2008) by Massimiliano Dominici.

java. *Java Technology*. `http://java.sun.com`, March 2008.

Dave Pawson. XSL-FO. O'Reilly & Associates, Inc., August 2002.

Erik T. Ray. *Learning* XML. O'Reilly & Associates, Inc., January 2001.

Unicode Bidirectional Algorithm. The Unicode Consortium, `http://unicode.org/reports/tr9/`, March 2008. Unicode Standard Annex #9.

W3C. XSL *Transformations (*XSLT*). Version 1.0*. `http://www.w3.org/TR/1999/REC-xslt-19991116`, November 1999. W3C Recommendation. Edited by James Clark.

W3C. *Extensible Stylesheet Language (*XSL*). Version 1.0*. `http://www.w3.org/TR/2001/REC-xsl-20011015/`, October 2001. W3C Recommendation. Edited by James Clark.

W3C. *Extensible Stylesheet Language (*XSL*). Version 1.1*. `http://www.w3.org/TR/2006/REC-xsl11-20061205/`, December 2006. W3C Recommendation. Edited by Anders Berglund.

W3C. XSL *Transformations (*XSLT*). Version 2.0*. `http://www.w3.org/TR/2007/WD-xslt20-20070123`, January 2007. W3C Recommendation. Edited by Michael H. Kay.

W3C. *Design Notes for Extensible Stylesheet Language (*XSL*) 2.0*. `http://www.w3.org/TR/2009/WD-xslfo20-20110927/`, September 2011. W3C Working Draft. Edited by Dave Pawson.

▷ Jean-Michel Hufflen
LIFC
University of Franche-Comté
16, route de Gray
25030 BESANÇON CEDEX
FRANCE
`jmhufflen at lifc dot univ-fcomte dot fr`

Questa rivista è stata ristampata
presso Braille gamma S.r.l., Cittaducale (RI)
su carta Revive 100 uncoated natural 80 g
distribuita da Polyedra.

ArsTEXnica – Call for Paper

La rivista è aperta al contributo di tutti coloro che vogliano partecipare con un proprio articolo. Questo dovrà essere inviato alla redazione di ArsTEXnica, per essere sottoposto alla valutazione di recensori entro e non oltre il 14 Febbraio 2012. È necessario che gli autori utilizzino la classe di documento ufficiale della rivista; l'autore troverà raccomandazioni e istruzioni più dettagliate all'interno del file d'esempio (.tex).

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

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

Chiunque, poi, volesse collaborare con la rivista a qualsiasi titolo (recensore, revisore di bozze, grafico, etc.) può contattare la redazione all'indirizzo `arstexnica@sssup.it`.

ArsTEXnica

Numero 12, Ottobre 2011

5 Editoriale
Gianluca Pignalberi

6 XeLaTeX and the PDF archivable format
Claudio Beccari

12 Creare grafici con pgfplots
Agostino De Marco, Roberto Giacomelli

39 LaTeX nella pubblica amministrazione.
La web application per la produzione automatica di comunicazioni interne del Comune di Napoli
Agostino De Marco, Paolo Eugenio Cresci

57 Presentazioni da ArsTEXnica numero 11

58 An Introductive Presentation of XSL-FO
Jean-Michel Hufflen

20012
9 771828 236001