New Developments in Electronic Publishing
AMS/SMM Special Session, Houston, May 2004
ECM4 Satellite Conference, Stockholm, June 2004
p. 81-90.

Some Remarks on the New Schemes in Electronic Publishing in Mathematics

Constancio Hernández[*]

Departamento de Matemáticas, Universidad Autónoma Metropolitana
Av. San Rafael Atlixco # 186, Col. Vicentina, C.P. 09340, Iztapalapa, Mexico, D.F.
e-mail: chg@xanum.uam.mx

Abstract. We analyze the emergence of new models of e-publishing from the experience of e-archives and independent journals. In the next few years the power and availability of resources, in hardware and software, to create and disseminate publications will increase. So, the schemes in journal publishing are changing and will change more. Journals face the problem of enhancing their quality and adopting standards that make access to scientists easier today and in the future. We will present some considerations regarding these issues.

Keywords: Portable Document Format, Mark up Language, TEX, HTML, Journal Publishing

1. Introduction

The ways in which mathematical research is being communicated throughout the world and the nature and cost of scientific journals is changing, and will change more. This article examines this issue from the point of view of a mathematician who has been actively involved in the journal editorial process.

The publishing world is moving rapidly into the digital age and the new Web technology could make the communication and dissemination of information much faster and efficient. Another important effect of the new technology on the publishing industry is the reduction of costs that once were unavoidable. All that represents an opportunity and a challenge for nonprofit periodic publications, traditional or electronic (also called independent journals).

[*]The research is partially supported by Consejo Nacional de Ciencias y Tecnología (CONACyT), grant no. 400200-5-38164-E

Most independent journals also face severe financial constraints, and the electronic dissemination via the Web is not only an additional medium, but a necessity in order to publish and participate in sharing the information and benefit from the wealth and progress it creates.

One important problem that remains to be solved is quality, both in content and form. The quality of the contents can be guaranteed by adopting the known process of blind-refereeing, editing, and accepting/rejecting, building upon the expertise of many of the existing publishing houses. The quality of the presentation is a problem that has not been fully solved yet. There have been many public debates about which is the best format to present faithfully the visual information inherent in mathematical or chemical formulae keeping, at the same time, the flexibility of interactive hypertext searching possibilities.

TEX is a system that fits perfectly into the picture of producing scientific documents of the highest possible typographic quality to make them available in the Web. Moreover, since almost all mathematicians produce an electronic version of their research papers in some flavour of TEX (most of them do it themselves, but some get help from another person in their department or institution), the Web is no threat to LATEX but an additional opportunity. We expect that scientists will continue to use the tools they feel most comfortable with for writing their documents and make their results available to everyone.

The TEX community has known the concept of cross-references well for long time, and the open architecture of the `\special` command makes it relatively easy to translate the cross-reference information to hypertext linking commands. We can also extend viewers to use these link commands and to use them as simple hypertext browsers. There are several packages that already do this, like `hypertext` for instance.

If we use TEX, an easy way to put a scientific document with the highest typographic quality in the Web is to make available the DVI produced by TEX. Another simple approach that guarantees typographic quality to the same level as that of a DVI file, is to generate a PDF file. For that purpose, we can use `dvipdfm` or pdfTEX or Acrobat Distiller starting from a PostScript file. In this case, with the help of a PDF browser, such as Acrobat, we can easily navigate through the document and exploit hypertext information that might be present in the LATEX source (LATEX cross-references and bibliographic citations can be automatically turned into hyperlinks). Web browsers can be configured to load as a helper application for PDF files, and they will then display our document in a browser window. Acrobat can also pass URL links to the browser to resolve, giving a seamless integration.

We might also decide to choose HTML as the format for our document in the Web. We need to translate our LATEX source directly into HTML and there are tools for that purpose such as LATEX2HTML or TEX4ht. But rendering mathematical and other technical notation is *not* an easy task. Mathematical notation not only involves the use of a large symbol set, it relies on a wide collection of 2-dimensional layout constructions, such as exponents, fractions, radicals and matrices, just to mention the most common. The way used by LATEX2HTML or TEX4ht to overcome some of the inadequacies of HTML for technical documents is to transform non standard characters (Greek, mathematical symbols, and so on) into bitmap pictures. The information is correctly displayed, but we must be careful to tune the visual representation (size, style, and so on) of the special characters so that their bitmap images in running text and mathematical displays match. However, since the user can change the default font and size of the normal text in his browser, it is practically impossible to ensure that everyone sees the "right" result. In addition, equations cannot "reflow" when the window size changes since they are fixed-size pictures.

Working with these systems, we can obtain an acceptable quality with a standard browser. Moreover, the cross-references, bibliographic citations, and hierarchical structure of the LaTeX source file can be translated into HTML hypertext functionality, allowing for optimal navigation and integration in the Web. Far-reaching customization is possible via command option and extention files.

On the other hand, because most mathematical symbols are translated into bitmaps, many thousands of bitmaps images may be created for a scientific document and these files have to be downloaded together with the HTML source of the page. This can make the time needed to transmit through internet and display a page in a browser rather long. These images also take a lot of diskspace, and often a modification of the LaTeX source needs rebuilding the whole set of images. Finally, the installation of a LaTeX-to-HTML translator is not simple.

If we want to put a document in the Web, starting from TeX, for example, we have various formats, and ways to transform between them, in which the document can be prepared for that purpose, as shown in Figure 1. We start with a LaTeX document. Of course, we can make available the LaTeX document itself, but it is more convenient to transform it into a format standard for that medium. A well-known procedure is to compile the LaTeX file with TeX, generating a DVI file, and from there to obtain a PostScript file with `dvips.` The latter file can be printed, if needed. Alternatively, you can generate a PDF file, directly from LaTeX with pdfTeX or indirectly by translating the PostScript file into PDF (for example, with Adobe's Acrobat Distiller) or by translating the DVI file into PDF. The PDF file can still be printed, if needed, but it is equally convenient to make it available on the internet, since it can contain hyperlinks and is editable.

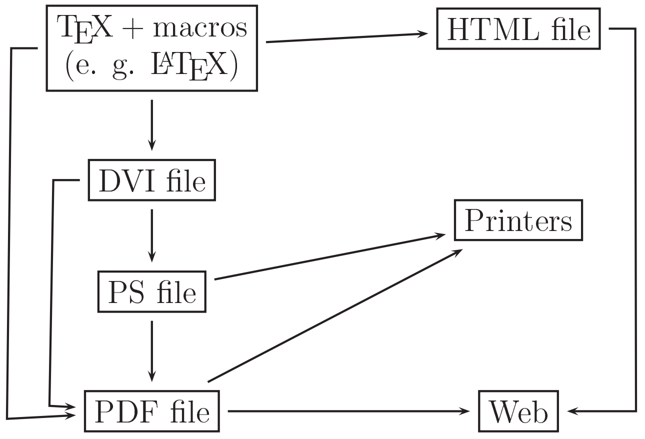

Figure 1

Another approach is to enhance the browser with a plug-in. For example, the plug-in `techexplorer hypermedia,` originally developed by IBM and now available from Integre Technical Publishing Co., allows your browser to display TeX, LaTeX, and MathML markup, either from separate source files, or embedded into an HTML page. The drawback is that the `techexplorer` plug-in must be downloaded and installed on the browser you want to use, and a different version is needed for each computer platform.

Finally, we can translate our LaTeX source into the MathML language, but in this case we must have access to a browser plug-in or some other method that can parse and display MathML. An option, based on Java applets, is WebEQ which can produce good quality display. Its more important advantage is that it can adapt to resizing of the browser window size and fonts. This solution, however, can be time-consuming because the wait for the initial download of the Java applets can last well over half a minute. Of course, we can install the Java code on our machine so that it is always available; even so, we have to wait while equations are processed. Another problem with WebEQ is that it does not strictly process LaTeX, but rather a variant called WebTeX.

The optimal approach depends on the nature of our document and its purposes. In deciding which method is most suited in a given situation, keep in mind the following considerations:

1. If the highest quality is essential or your document has a complex layout (elaborated mathematic constructions, tables) or typography, use PDF.
2. If it contains lots of normal math, then you would probably want to translate it into MathML, and use one of the browser plug-ins.
3. If your document contains low-level math with macros that allow you to use your own customized notations, then use TeX4ht or LaTeX2HTML.
4. If the document is purely text, and straightforward LaTeX, then translate it directly into HTML, or into XML.

2. Portable Document Format

In the case of journal publishing the quality is essential and the methods to disseminate information across the Web that leaves the rendering of the "page" to the readers software are unacceptable. When we elaborate a document using the traditional markup languages of the Web, it is almost impossible to ensure that the reader sees the "right" result. We cannot control how the lines break in a paragraph and, of course, there is no concept of page break. Graphics are often presented as low resolution bitmaps with unreliable colors; table layouts may seen radically different. Finally, and perhaps most important, the current generation of Web browsers is not very sophisticated at typesetting and page makeup; the results of printing a document from a browser does not produce a high-quality result.

Mathematics is a typesetting challenge and TeX allows for setting professional-quality mathematics. From a precise knowledge of the sizes of all characters and symbols, the TeX system computes the optimal arrangement of letters per line and lines per page (this is the typesetting process). Then, it produces a DVI file (device independent) containing the final locations of all characters. In other words, the DVI file contains the description of the page character by character, and it is encoded in a language that can be translated to any printer or display device language. We can make the DVI file available in the Web, or we might also, instead, decide to translate it to the PDF format.

2.1. What is PDF?

Portable document format, PDF, is a document file format proposed by Adobe Systems for the distribution and exchange of electronic documents and forms around the world. It is a second-generation page description language descendant from PostScript. Adobe PDF has the same imaging capabilities as PostScript because they share the underlying architecture of Adobe Imaging Model. However, there are some important differences between them:

- PostScript is a full-blown programming language, but PDF is not. (Adobe added Java Script support in version 3.5 of the Acrobat Forms plug-in, but this has more to do with viewer than with the PDF language.) This difference – while important – is meaningful only to very few people.
- The structure of PDF files is much more predictable than the structure of PostScript files. This is important in situations where software needs to modify a document or extract information from a document. In particular, the PDF file guarantees page independence, clearly separating resources from page objects.
- A PDF file can contain links to locations within the same PDF file, within other PDF files, or on the Web, that is, the PDF language has hypertext capabilities; a PostScript file normally does not contain links.
- A PDF file can function as a data-entry form, but a PostScript file cannot.
- In general, a PDF file is smaller than the PostScript version file. This is important, because smaller files can be handled more efficiently (it takes less time to send smaller files across a network).

Most of the advantages of PostScript remain: PDF preserves the fonts, images, graphics, and layout of any source document, while being portable across different computer platforms.

For these reasons and more, PDF is becoming a replacement for PostScript in many situations.

It is important to make a clear distinction between Portable Document Format, and Adobe Acrobat: PDF is an open file format specification (although Adobe controls it), and is available to anyone who wants to develop tools to create, view, or manipulate PDF documents; Acrobat is a family of commercial programs from Adobe which produces, displays, and manipulates PDF. The main components are: Reader, distributed freely, for viewing and printing PDF files; Exchange, which has all the functionality of Reader, but allows changing the file; Distiller, which produces PDF from PostScript files; PDFWriter, which provides printer drivers for Windows and Macintosh, allowing any application to "print" to a PDF file directly; Catalog, which makes indexes of collections of PDF documents; and Capture, which produces PDF files by performing optical character recognition on bitmap-scanned pages.

Acrobat Reader is free and available on almost all platforms. However, there are other viewers available as well. Ghostscript and Xpdf are the best-known and are free. Ghostscript also produces PDF from PostScript.

2.2. How can we create a PDF file from a TeX source?

A PDF file can be created in four basic ways: convert an existing PostScript file to PDF using a translation program. The Adobe Acrobat Distiller is the most powerful of these, but Ghostscript also performs well. This approach means that you can create PDF from any application that can produce PostScript (that is, almost everything); use Adobe's PDF Writer printer driver for Windows and Macintosh to produce PDF from any normal application like a word processor; use an application that writes PDF directly, for example, TEX users have pdfTEX, VTEX, and the DVI driver dvipdfm; use Adobe's Acrobat Capture software to convert, using a scanner, large volumes of paper documents into a PDF workflow. There is a feature by which words that cannot be recognized are preserved as bitmaps.

Since almost all text formatting software can write good PostScript, the most common method is the first. The second is the quickest and easiest method for creating PDF files, but because it doesn't give opportunity to add hypertext links automatically and gives no control over features like compression and sampling of art work, it is not recommended for serious work. The third method is, naturally, the ideal one, but there are few examples of suitable applications. The last method is rather specialized and is really suitable only for large-scale projects converting legacy documents with experienced staff controlling the quality.

2.2.1. Setting up fonts

Documents created in Windows applications may contain Type 1 fonts, Type 3 fonts, Windows bitmap and vector fonts, TrueType fonts, and Open Type Fonts (Windows 2000 and later). Documents created in Mac OS applications can contain Type 1 fonts, Type 3 fonts, Mac OS bitmap fonts, and TrueType fonts.

A PDF-producing application can handle fonts by doing one or more of the following:

1. *Adding a description of the fonts* containing information such as name, metrics, encoding, type (sans serif, symbol, for example) and clues about its design, and rely on the display application to show something plausible. This method creates the smallest files. The display application can work again in several ways. It can try to find the named fonts on the local system; it can simply substitute system fonts as intelligently as possible; or it can use Multiple Master fonts to resemble the appearance of the original font.
2. *Embedding the fonts.* A font that is embedded in a PDF file is always available for viewing and printing, whether or not it is installed on the system.
3. *Subsetting the fonts,* that is, embedding only the characters of a specific font that are used in a document. By subsetting fonts, the printer driver decreases the file size, which in turn increases the speed of PostScript file conversion.

TEX systems have traditionally depended on the use of fixed resolution bitmap (that is, .pk) fonts. In principle, given adequate resolution, the screen preview quality of documents set in bitmap fonts, and set in outline fonts, should be comparable, since the outline fonts have to be rasterized dynamically anyway for use on a printer or a display screen. Sadly, this is not the case with Adobe Acrobat version 5 or earlier; they do a poor job of downsampling high-resolution bitmap fonts to low-resolution screen fonts. However, Acrobat 6 and that version of Reader now do a much better job with bitmap fonts.

If we want to produce good-quality PDF, it is crucial to overcome the problem of bitmap fonts in TEX output. So, we need to find outline versions (Type 1, True Type, Open Type, for example) of the fonts that we intend to use and then configure the driver so that it uses them. How this is done depends on the DVI driver. A very good treatment on using PostScript outline fonts with TEX can be found in the excellent book from Goossens, Rahtz, and Mittelbach [8]:

For instance, Y&Y's dviwindo and dvipsone drivers, support only Type 1 fonts and can access whatever is installed in Adobe Type Manager. The most used dvips uses a mapping file, psfonts.map, to relate TeX font names to physical file names found in the font search path. Typical entries in this file look like this:

```
cmr10   cmr10                                 < cmr10.pfb
rptmb   Times-Bold
rptmbo  Times-Bold    ".167 SlantFont"
lbr     LucidaBright                          < lbr.pfb
logo10  logo10                                <logo10.pfb
logo8   logo8                                 <logo8.pfb
```

The first entry on each line is the name of the TEX Font Metric (.tfm) file; the second is the PostScript font name; the remaining entries may be a quoted string containing driver-specific PostScript code for modifying the font, or a left-angle-bracket-prefixed PostScript font file name, generally ending in .pfb (PostScript Font Binary). This last item tells the DVI driver that the font must be included in the PostScript file. Otherwise, the font is assumed to be printer-resident.

Normally, psfonts.map does not contain entries for the standard Computer Modern family, because not all sites have outline versions installed.

Modern TEX installations include a MakeTeXPK script, or its equivalent, that DVI drivers invoke to create a missing font on-the-fly. That script examines the psfonts.map to see if the requested font is specified there. If so, it invokes a font rasterizer, such as gsftopk or ps2pk, to produce a standard TEX PK-format bitmap font. Otherwise, it invokes Metafont to generate a GF-format bitmap font, and then runs gftopk to convert that to a more compact PK-format bitmap font.

The normal DVI drivers default to using bitmap fonts. Thus, to make your TEX DVI driver use outline fonts instead of bitmap fonts, you need to make sure that psfonts.map lists all of the fonts that you want to use in outline form.

There are two available sets of Computer Modern outline fonts, the BaKoMa family, and the BlueSky Research/AMS family, both available in the Comprehensive TeX Archive Network (CTAN) under the directory /tex-archive/fonts/cm/ps-type1. In this archive, we can also find some other public domain technical fonts that have been converted to Type 1 format like LATEX additions of the Computer Modern family; the American Mathematical Society fonts; the St. Mary's Road Symbol fonts; the RSFS script font; the TIPA phonetic fonts; the XY-pic fonts.

There are some other alternatives of outline fonts for TEX, for example, the LucidaBright Family from Bigelow & Holmes. The Lucida family is well-known through its use for the last years in Scientific American. It was initially designed specifically for high legibility on low-resolution printing devices, and consequently, its lower-case letters are somewhat larger than in other fonts. Another example are the Type 1 and True Type versions of the "European Computer Modern" (ec) fonts from Micro Press. We have also two other versions: the ae package provides virtual fonts that match the ec fonts as much as possible and draws on the original Computer Modern fonts. There are a few missing characters, like gillemets; the commercial European Modern font set by Y&Y is a set of high-quality, fully hinted fonts that can fully replace ec.

Finally, if you use Acrobat Distiller, the most reliable way to include fonts is to embed them in the PostScript file before distilling. This is the only way to include TrueType fonts. If Distiller cannot locate PostScript fonts, you may have to use the Font Locations menu option to tell Distiller where the fonts are located. This is under the Settings menu to Font Locations. Acrobat 4 and above honor licensing agreements for TrueType fonts and may not let you embed copyprotected fonts. If the fonts are failing to embed due to licensing restrictions this will show up in the Distiller dialog box and will be recorded in the log file. This can happen with some TrueType fonts, foreign language fonts and fonts created with Fontographer.

2.3. Hypertext links and other features

PDF format offers several possibilities for navigation: bookmarks for the logical view, hypertext links for switching between various physical or logical views and, in the last versions, the possibility of interaction with other applications. With the Web, this kind of features has become the paradigmatic structure of a distributed archive. Making an electronic document that takes advantage of all that, and not only a PDF image of the normal printed page, is very convenient for almost all applications. At a minimum, cross-references need to be added in our PDF in the form of hypertext links. However, many people also expect the possibility of automatic bookmarks, that URLs be active links, and the possibility of adding new arbitrary links. Of course, we can add the links in Acrobat Exchange manually, but this is error prone and has to be repeated each time the document changes. Acrobat Distiller recognizes a special PostScript command, `pdfmark`, that can be used as a hook to insert a vast amount of functionality into a PDF file. Thus, we need that our application embeds that special code in the PostScript file. In the particular case of the TEX users, it is rather easy since almost all DVI PostScript drivers allow for the insertion of raw PostScript into the output file. For instance, `dvips` recognizes the form `\special{ps:: ...}` to insert any PostScript code you like.

However, you dont have to write a line of PostScript code because there are macro packages that translate the cross-referencing mechanisms and other features of LATEX (like mark up for chapters, sections, etc.) to the necessary `pdfmark` commands, that in turn, will be converted into hypertext links and bookmarks by the PDF creator program (Acrobat Distiller or GhostScript). The best known of these is the `hyperref` package by Sebastian Rahtz and Heiko Oberdiek. Other options of this kind of packages are: `hyper`, by Michael Mehlich; ConTEXT, by Hans Hagen; `hyperbasics.tex` by Tanmoy Bhattacharya (this is a simple example of how to write a macro package supporting HyperTeX).

The `hyperref` package derives from the HyperTEX project [10] and extends the functionality of all LATEX cross-referencing commands via `\special` commands in such a way that any DVI driver can turn them into hypertext links and not only for rich PDF generating purposes.

Finally, if our application generates PDF code directly we have, perhaps, the best solution. In TEX world, we have pdfTEX that can produce PDF output instead of the standard DVI file and provides access to all features of PDF. The pdfTEX program adds a number of primitives to the TEX language that can be used directly. In this case, we have the hypertext information in the source.

3. Conclusion

An electronic journal needs to choose a file format for the storage and the exchange of documents and the minimal properties and capabilities of this file format must include:

- Typographic quality in printing;
- Portability;
- Hypertext;
- Simple, intuitive, aesthetic, and highly documented specification;
- Support of metadata;
- Compression.

The most important options for a technical journal are DVI, PDF and MathML. The first two are page description languages and, because they are binary encoded, are not human readable. For this reason, we can not talk about an intuitive and aesthetic specification in the case of PDF and DVI. Moreover, we need another application that generates the PDF or DVI from a source in another language. For PDF, the source could be in TEX or in MathML (for DVI only in TEX). Programs like TEX are page construction applications (and the quality of TEX is equivalent to the highest in typography). Documents in DVI format are not really portable because of `\special` commands that it could contain. Finally, DVI does not support metadata, and only versions 5 and 6 of Distiller support the embedding of XMP metadata in PDF. On the other hand, the third, MathML, is a markup language and its use is in its beginnings as is the development of software that supports the language. Making available a MathML file in the Web is equivalent to making available a TEX or HTML file in the sense that we leave the rendering of the page to the readers software with all that this implies. However, I think that MathML is the next step in online mathematics for its capabilities of computational interaction with other applications regarding mathematical expressions. With MathML, a user will be able to copy/paste a mathematical expression found anywhere on the Web into a computer mathematics system and have a reasonable expectation of being able to evaluate it. We must then decide if that feature is important for our documents.

For these reasons, MathML could be the best option in the short term, but at the moment PDF is a good choice at least for the distribution of electronic documents.

References

[1] Adobe System Incorporated: *PostScript Language Tutorial and Cookbook*. Addison-Wesley, Reading 1985.

[2] Adobe System Incorporated: *PostScript Language Reference Manual.* Addison-Wesley, Reading 1990.

[3] Adobe System Incorporated: *Adobe Type* 1 *Format.* Addison-Wesley, Reading 1990.

[4] Bienz, T.; Cohn, R.; Meehan, J. R.: *Portable Document Format Reference Manual Version* 1.2. San Jose, Calif., Adobe Systems Inc. 1997. Available online at `http://www.baw.de/vip/abteilungen/wbk/Publikationen/docs/pdfspec.pdf`.

[5] Carr, L.; Rahtz, S.; Hall, W.: *Experiments with TEX and hyperactivity.* TUGboat **12**(1), 13–20.

[6] Flynn, P.: *Math on the Internet.* IEEE Spectrum, **36**(4), 36–40.

[7] Goossens, M.; Mittelbach, F.; Samarin, A.: *The LATEX Companion.* Addison-Wesley Inc., Reading Mass. 1994.

[8] Goossens, M.; Rahtz, S.; Mittelbach, F.: *The LATEX Graphics Companion.* Addison-Wesley Inc., Reading Mass. 1997.

[9] Goossens, M.; Rahtz, S.: *The LATEX Web Companion.* Addison-Wesley Logman Inc. Reading Mass. 1999.

[10] HyperTEX FAQ, `http://xxx.lanl.gov/HyperT\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\spacefactor\@m/`.

[11] Knuth, D. E.: *The TEXbook, volume A of Computer and Typesetting.* Addison-Wesley Inc., Reading Mass. 1986.

[12] Lamport, L.: *LATEX: A Document Preparation System: User's Guide and Reference Manual.* 2nd edition, Addison-Wesley, Reading Mass. 1994.

[13] Merz, T.: *Web Publishing with Acrobat/PDF.* Springer-Verlag, Berlin 1998.

[14] Mittelbach, F.: *E-TEX: Guidelines for future TEX extensions.* TUGboat **11**(3) (1990), 337–345.

[15] Mittelbach, F.; Rowley, C.: *LATEX* 2.09 $\rightarrow$ *LATEX*3. TUGboat **13**(1) (1992), 96–101.

Received September 27, 2004