Multipub: managing a LaTeX book manuscript for concurrent print and electronic formats

Paolo Prandoni

November 26, 2015

1 Introduction

So, you just put the finishing touches to your LaTeX manuscript, which compiles to a beautiful camera-ready PostScript file: your book is finally complete! ...or is it? Aren't you forgetting about the *electronic version* of the book?

If you are a novelist, no problem, there are plenty of tools to convert “pure text” into virtually any ebook format. Authors of technical and mathematical texts, however, are faced with the challenge of mathematical formulas and graphic illustrations. The problem with formulas is due to the reflowable nature of ebook content: when the electronic format has a pre-arranged, fixed layout (as in PDF documents, for instance), equations can be easily rendered as images and everything looks nice; but when the document or the font sizes are allowed to change, a mechanism to render the math on the fly becomes necessary. In the past, when reflowable content basically meant HTML pages, the de-facto solution was to pre-render all mathematical formulas as bitmap images; however this creates numerous problems, from inline formulas that do not align properly to equations that do not rescale when the surrounding font size is varied.

Pre-rendering is also necessary if you have defined your illustrations in PostScript within your manuscript; since figures are floating freely within the page, this is much less of a issue (although resizing may still be problematic).

While pre-rendering of images can be acceptable, pre-rendering of math is not an option, at least in my opinion; I want fully reflowable math in my ebooks.

In the rest of this memo I will detail the steps I put in place to manage the concurrent PDF and eBook rendering of a LaTeX manuscript. Starting with a document in LaTeX, the objective is to have an efficient workflow that, based on a *single* master manuscript, can produce both a PDF and a reflowable version of the text. Changing the compilation target should not require any manual change in the main text other than choosing the compilation target; obviously, different style files can (and shoud) be included according to the target.

To recap, the goals of the toolchain are:

- compilation to PDF format for printing
- compilation to electronic format with MathML support for modern e-readers and web browsers.

2 Choosing the Links in the Toolchain

Executive summary:

- EPUB is the chosen eBook format
- MathML will be used for formulas
- LateXML will be used for EPUB compilation

The longer story is like so:

2.1 Render Mathematical Formulas and eReaders

Bitmap formulas look awful, so the only two approaches to reflowable, good quality math available today for browsers and eReaders are MathML and MathJax. MathML is a standard supported natively by most browsers (although, surprisingly, not by Chrome) in which equations are encoded in XML and rendered on the fly. The second approach, MathJax, allows for math to be written in a variety of for-

mats (including LaTeX and MathML itself) and the rendering is performed by Javascript; obviously, this requires a browser (or a reader) equipped with Javascript and sufficient computational power. MathML is "understood" by MathJax, so I chose **MathML**.

2.2 eReaders and eBook Formats

Today, the most common devices to read eBooks are:

1. Kindle Paperwhite (or similar e-paper devices)
2. iPad or Android tablets
3. PCs

On the other hand, there are about *thirty* eBook formats in use worldwide. Many of these are either specialized formats (e.g. comic books) or proprietary formats (e.g. Sony reader format). The most important items for our purposes are:

- EPUB: an open standard now in its 3rd version based on HTML+CSS. Supports

MathML and Javascript.

- iBook: a non-standard EBOOK subset by Apple; not compatible with EPUB readers. Supports MathML and Javascript.
- AZW: Kindle's proprietary format (now in version 8). Does NOT support MathML.
- PDF: although not an ebook standard per se, still very relevant. Supports Javascript.

Each standard should be evaluated according to two independent criteria: how beautiful the books look and what devices these book can be read on; additionally, one should consider DRM and marketing channels.

- iBook produces very high quality books since the books only work on Apple devices and therefore the environment is fully controlled; however, books only work on Apple devices.
- AZW works natively on Kindle but Amazon has Kindle reader software for all platforms (including Apple devices)
- EPUB is an open format and readers exist for tablets and PCs (but not for the Kin-

dle). However, your mileage may vary with each reader since the standard is open and potentially very broad.

- PDFs are supported by all devices but reflowing is not possible.

AZW (the Kindle format used by the "classic" Kindles) does not support MathML so equations must be rendered as graphics. However, and this is critical, the AZW format *does not support inline graphics*, so inline equations look terrible. For classic Kindles, the best approach is simply to produce a fixed-layout PDF with a size compatible with the Kindle screen. (This is what Amazon does already for existing textbooks).

iBook is a closed standard and Apple handles the sales channel with an iron fist. Since iBook is basically a proprietary dialect of EPUB and since excellent EPUB readers exist for iPad, iBook would be difficult to recommend over EPUB.

EPUB therefore seems to be the obligatory choice although, being an open standard, one should not expect the flawless level of device support offered by the proprietary formats.

2.3 LaTeX Compilation

Since the chosen target format, EPUB, is based on XHTML, the first order of business is finding a good tool to convert LaTeX to HTML. Since TeX is a Turing-complete programming language, conversion is anything but trivial and, indeed, independently of the chosen tool, there will be limitations on the external packages that can be used in a convertible a manuscript.

Amongst the available conversion tool, I chose **LaTeXML** [**?**] since it formats equations in MathML and it supports a reasonable number of external packages via so called “bindings” (see Figure 1 and [**?**] for the up-to-date list). A binding is a piece of custom code that makes the package understandable by LateXML; writing a custom binding seems to be a daunting task so we will not attempt that here.

In Section **??** I will describe how to install the full toolchain; a shell script for one-line compilation of the manuscript for different targets is also provided.

3 Preparing the Manuscript

The `multipub` package described in this section defines three compilation targets:

- PRINT: standard compilation leading to printable (PDF) output
- KINDLE: similar to PRINT, but produces a specially-formatted PDF to fit the screen of Kindle-like readers
- EPUB: compilation with LateXML for ebooks

These three targets can be additionally used to conditionally include specific packages and macros.

3.1 Conditional Compilation

Include[1] the `multipub` package right after the `documentclass` declaration in your master doc-

[1] I couldn't package `multipub` as a style file because LateXML only accepts style files for which it has bindings.

ument and specify your compilation target like so:

```
\documentclass[...]{...}
\input{multipub}
\multipub{TARGET}
```

where TARGET is one of the compilation targets defined above. The three compilation targets can be used to define target-specific portions of the LaTeX manuscript that will be active only when the target is selected; for instance

```
\begin{PRINT}
  % custom page formats for the PRINT version
  \include{styles/printlayout}
  \usepackage[utopia]{mathdesign}
  \usepackage{pstricks}
\end{PRINT}

\begin{EPUB}
  % custom page formats for the EPUB version
  \include{styles/epublayout}
\end{EPUB}
```

3.2 External Packages

The first step in preparing a manuscript for multi-target compilation is to **make sure that all critical packages have available LaTeXML bindings** (again, see Figure 1). A critical package is a functional package that will be needed for all compilation targets. Please note that any package affecting the page layout only (e.g. specific font packages) will be irrelevant when compiling for the EPUB target and so it can be conditionally included as explained above. Another exception is the inclusion of PSTricks-derived packages, for which a different strategy is provided.

3.3 File Structure for Multi-part Documents

With multi-part documents (such as the chapters in a book or different sections in a longer text) it is often convenient to have a master document and include the relevant files from subfolders; e.g. if you have a `book.tex` master file in the top-level directory and a series

a0size, a4, a4wide, aasms, aaspp, aastex, acronym, ae, afterpage, alltt, amsbsy, amscd, amsfonts, amsmath, amsopn, amsppt, amsrefs, amssymb, amstex, amstext, amsthm, amsxtra, apjfonts, array, avant, babel, bbm, bbold, beton, bm, bookman, booktabs, braket, calc, cancel, caption, ccfonts, chancery, charter, circuitikz, cite, citesort, cmbright, color, colordvi, colortbl, comment, concmath, courier, crop, cropmark, dcolumn, deluxetable, doublespace, dsfont, ellipsis, elsart, emulateapj, emulateapj5, enumerate, epigraph, epsf, epsfig, epstopdf, eucal, eufrak, euler, eulervm, eurosym, euscript, exscale, fancyhdr, fix-cm, fixltx2e, flafter, fleqn, float, floatfig, floatflt, floatpag, fontenc, fontspec, footmisc, fourier, framed, fullpage, geometry, german, graphics, graphicx, grffile, helvet, here, hhline, html, hyperref, hyperxmp, ifpdf, ifthen, ifvtex, ifxetex, import, indentfirst, inputenc, iopams, keyval, latexml, latexsym, lineno, listings, listingsutf8, longtable, lscape, luximono, lxRDFa, makeidx, mathpazo, mathpple, mathptm, mathptmx, mathrsfs, multicol, multido, multirow, nameref, natbib, newcent, ngerman, nicefrac, ntheorem, numprint, palatino, paralist, parskip, pdflscape, pdfsync, pgf, pgfplots, pifont, placeins, preview, psfig, pslatex, pspicture, pst-grad, pst-node, pstricks, pxfonts, relsize, revsymb, revtex, revtex4, rotate, rotating, rsfs, scalefnt, setspace

of subfolders containing chapters you may find yourself writing something like:

```
\documentclass[...]{book}
\begin{document}

\title{My Book}
\maketitle

\include{ch01/chapter1.tex}
\include{ch02/chapter2.tex}

\end{document}
```

The rationale behind this file structure is to organize the material for each chapter in separate subfolders, possibly creating another level of folders in each subfolder to collect figures or data files. The problem, however, is that if you now, say, include a figure in a chapter file using a path relative to the chapter's subfolder, the figure will not be found since LaTeX assumes that all paths are relative to the master document. To work around this problem without using absolute paths, `multipub` defines two specific macros. In your master document, use the `\includefile` macro to include chapters, sep-

arating the path from the file name:

```
\documentclass[...]{book}
...
\includefile{ch01}{chapter1.tex}
\includefile{ch02}{chapter2.tex}
```

then, in each included file, you can use the `\localpath` macro to include files relative to the included file itself. For example, assume the folder `ch01/figs/` exists; then in `chapter1.tex` you can use things like:

```
\includegraphics{\localpath{figs/fig.eps}}
```

and the document will compile. Using the `\includefile` macro is particularly useful with respect to PSTricks figures, as explained in the following section.

As a final note, *do NOT use underscores in the filenames or path names* as this seems to confuse LateXML.

3.4 Figures and PSTricks

LateXML supports picture inclusion via `\includegraphics`; in other words, figures defined as in the code snippet below should present no problem across compilation targets:

```
\begin{figure}
  \includegraphics{figs/figure.eps}
\end{figure}
```

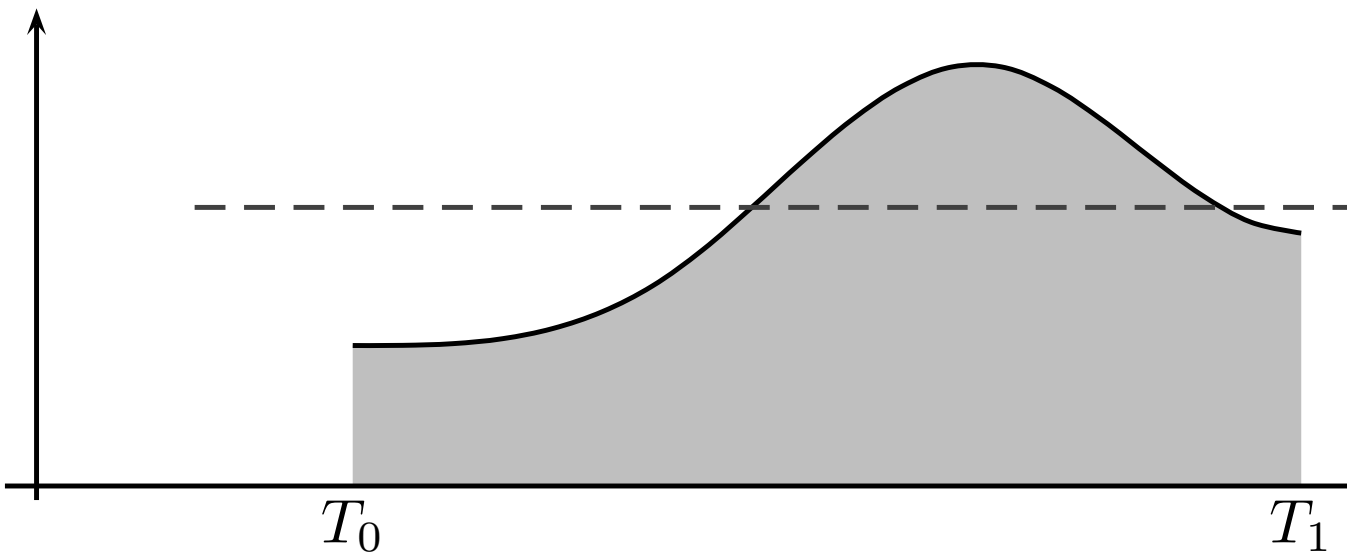
On the other hand, although LateXML has some basic PSTricks bindings, it certainly does not cover the multitude of useful PSTricks-based packages in use today. Therefore the strategy for cross-platform compilation is the following:

- in the main document folder (or in each chapter folder) create a subfolder called `ps`
- isolate the PSTricks-specific code for each figure in the tex file and save it in separate small file in the `ps` folder (e.g. `ps/fig3.tex`)
- use the `\pstf` macro to include the

PsTricks figure using the file name without extension

- before compiling for the EPUB target, pre-render all figures using the provided script

For example, consider the following figure:



obtained with the following code:

```
\begin{figure}
  \center
  %%%% BEGIN PSTricks %%%
  \savedata{\smooth}[
    1.0000 1.0000 1.1579 1.0011 1.3158 1.0130 1.
    1.6316 1.1201 1.7895 1.2408 1.9474 1.4232 2.
    2.2632 1.9852 2.4211 2.3076 2.5789 2.6068 2.
    2.8947 2.9824 3.0526 2.9828 3.2105 2.8418 3.
    3.5263 2.3338 3.6842 2.0725 3.8421 1.8769 4.
```

```
  \psset{xunit=1.8cm,yunit=0.8cm,linewidth=0.8pt
  \begin{pspicture}(-.5,-.5)(5.5,3.5)
    \pscustom{
      \dataplot[plotstyle=curve,linewidth=1.1pt]
      \gsave
      \psline(4,0) \psline(1,0)
      \fill[fillstyle=solid,fillcolor=lightgray]
      \grestore}
    \psline{->}(0,-.1)(0,3.4)
    \psline{->}(-.1,0)(5.4,0)
    \rput[t](1,-.1){$T_0$}
    \rput[t](4,-.1){$T_1$}
    \psline[linestyle=dashed]{-}(0.5,1.983)(4.5,
  \end{pspicture}
  %% END PSTricks %%%%
\end{figure}
```

Cut all the lines between the `BEGIN PSTricks` and `END PSTricks` markers and save them as the file `ps/fig1.tex`. Now you can include the figure with

```
\begin{figure}
  \pstf{fig1}
\end{figure}
```

When compiling for the PRINT target, the macro

will include the pre-rendered version of the figure if it is available, otherwise it will include the PSTricks code directly (obviously in this case the PRINT version must include all necessary PSTricks packages; you can do that within the conditional inclusion section in the preamble of the document). When compiling for EPUB the pre-rendered figures *must* be available and a compilation error is raised otherwise.

4 Compilation

4.1 Setting Up the Toolchain

If you start from a fresh Linux installation[2] (e.g. Ubuntu), first install the necessary LaTeX components:

[2] Unfortunately, LaTeXML is written in Perl; installing Perl under Windows is far from trivial and therefore the best way to handle multi-format compilation under Windows is to install a Linux virtual machine; VirtualBox [**?**], for instance, is a free and efficient solution that interfaces easily with the host OS. In the following, I will detail the installation process for the Linux environment only.

```
> apt-get install texlive
> apt-get install texlive-latex-extra
> apt-get install latexml
```

4.2 PRINT and KINDLE Versions

Compilation in these cases is a simple LaTeX→ dvips → ps2pdf sequence. Using the provided script, simply type

```
> mpbuild PRINT filename
```

to obtain the final PDF document.

4.3 EPUB Version

EPUB compilation takes place in two steps, first the LaTeX manuscript is converted to XML and then the XML is converted to HTML; both operation are performed by LateXML. . In my case, the commands to perform the conversion are simply

```
> latexml --dest=book.xml book.tex
> latexmlpost --format=xhtml --dest=book.xhtml --p
```

The XHTML files can be browsed with a web browser or imported into an e-reader.

4.4 KINDLE Version

5 Windows Setup

sudo apt-get install texlive sudo apt-get install latex sudo apt-get install latex sudo apt-get install latexml cd book ls latex book.tex sudo apt-get install lighttpd