LaTeX at Distributed Proofreaders and the electronic preservation of mathematical literature at Project Gutenberg

Andrew Hwang

Abstract

A small but growing effort is underway at the volunteer web site Distributed Proofreaders (DP, at `www.pgdp.net`), with the goal of creating high-quality LaTeX files of selected public domain mathematical books for distribution by Project Gutenberg (PG). This article introduces DP and PG, describes how books are transcribed at DP, and gives an overview of current LaTeX coding strategies.

1 Introduction

Public domain mathematical works are a precious resource. Electronic preservation potentially makes historical mathematical literature available to anyone with a computer. By contrast, printed books and journals stored in university libraries suffer from constraints ranging from limited access to physical degradation.

This article describes a small but growing initiative to harness "crowdsourcing" for the purpose of transcribing public domain mathematical works into LaTeX. The existing web-based infrastructure is provided by Distributed Proofreaders (DP, at `www.pgdp.net`). The completed books are distributed by Project Gutenberg (PG, at `www.gutenberg.org`). The LaTeX work at DP and the availability of LaTeX source files for mathematical projects at PG are not widely-known. Please share this article with interested students and colleagues, and explore the sites yourself.

Since 2008, more than fifty LaTeX books have been produced at DP [1]. Recently-completed examples range in subject matter and sophistication from popular accounts to textbooks to research monographs. Titles include:

- *Mathematical Recreations and Essays* by W. W. Rouse Ball,
- *Philosophiae Naturalis Principia Mathematica* by Sir Isaac Newton,
- *A Short Account of the History of Mathematics* by W. W. Rouse Ball,
- *Calculus Made Easy* by Sylvanus P. Thompson.

The medium-term goals for LaTeX book production at DP are twofold: First, to recruit and build a community of LaTeX-knowledgeable volunteers; and second, to select and prepare suitable books from the mathematical literature of the 19th and early 20th Centuries. Further, DP can process any book for which copyright clearance is obtainable. Authors willing and able to grant perpetual, non-exclusive, worldwide rights to distribute their books in electronic form on a royalty-free basis can, at no cost to themselves, have their books converted to electronic form and made available at PG. A self-sustaining LaTeX community at DP stands equipped to generate a lasting scientific, cultural, historical, and educational resource.

2 Techniques of ebook production

Broadly speaking, "electronic preservation" may refer to anything from scanning a book and distributing bitmap image files (jpegs or pngs) to preparing an accurate, archival-quality textual representation, such as a well-designed LaTeX source file.

Scanning a book is relatively cheap and fast. A book of a few hundred pages can be scanned manually and non-destructively in about an hour by one individual without special skills or expensive equipment. Books can also be scanned destructively in bulk at high speed by cutting off the spine and running the pages through a mechanical feeder. At this writing and for the foreseeable future, the vast majority of mathematical ebooks consist of bulk-scanned images.

Once a book has been scanned, raw text may be extracted fairly easily with optical character recognition (OCR) software. Not surprisingly, however, mathematics is rendered poorly by OCR. As a result, raw OCR text of a mathematical book is all but unusable for a casual reader.

At DP, OCR text is the input material. Human volunteers carefully proofread the text against the page scans, then add LaTeX markup. The end result is an accurate textual and semantic representation of the book. Though producing a high-quality LaTeX source file requires on the order of an hour of skilled work *per page*, the benefits are substantial. For the typical reader, a LaTeX-produced PDF file is text-searchable, magnifiable on screen without loss of quality, easily-hyperlinked, and yields camera-quality printed output. To the benefit of readers without fast Internet access, a LaTeX-produced PDF file is about one-tenth the size of a collection of page scans; a compressed source file is smaller still. Thousands of textual books can be fit onto a DVD, compared with a couple hundred books made from scanned images. A good-sized library can therefore be easily and inexpensively distributed worldwide by ordinary post. Finally, if the coding is well-planned, a LaTeX source file can serve as an archival representation of the book.

2.1 Project Gutenberg and Distributed Proofreaders

Founded by Michael Hart at the University of Illinois in 1971, Project Gutenberg is the world's oldest electronic library. PG is dedicated to the storage and distribution of public domain ebooks.

Distributed Proofreaders was founded in 2000 by Charles Franks to produce ebooks for PG. The site source code, written in PHP, is free software released under the GNU GPL. The project homepage is `dproofreaders.sourceforge.net`. At this writing, there are at least six independent "DP sites" using some version of the code base. In addition to the DP site at `www.pgdp.net`, there are smaller "sister" DP sites based in Canada and Europe, which operate under the copyright laws of their respective regions. Due to lack of infrastructure and volunteers, LaTeX projects are currently processed only at `www.pgdp.net`, and the present article describes only activities at this DP site.

DP currently boasts a few hundred volunteers active on a near-daily basis, and produces a little over half of the new ebooks in PG's collection. At this writing, the number of volunteers who work on LaTeX is about 1% of the "population", and on average about 20 new LaTeX books are posted to PG every year.

The DP site at `www.pgdp.net` was designed and built entirely by volunteers, and is currently staffed by volunteers. DP-Canada, DP-Europe, and Project Gutenberg are also largely or entirely built and run by volunteers.

DP process overview

An ebook starts its life at DP as raw OCR output. The page-length pieces of OCR text and the page scans are loaded into a database hosted at DP. Working one page at a time, volunteers at the DP web site are presented with a scanned page image side-by-side with the corresponding OCRed text in an editor window. After correcting the text and adding LaTeX macros, proofreaders check the page back into the database. Once all the pages of a book have been reviewed and corrected, the site software concatenates the pages into a raw ebook file. A single volunteer performs final polishing and verification, then submits the completed ebook to Project Gutenberg.

The actual path of a book through DP is a bit more involved. The distributed work is separated into "proofing" and "formatting" stages. Proofing focuses on verifying the correctness of the raw words in the text, the general dictum being "match the scan". Because most DP volunteers do not speak LaTeX, the text file at the end of the proofing rounds omits most of the mathematical markup, and is far from being machine compilable. The formatting rounds add the necessary markup, including mathematics, footnotes, and sectional divisions. The output of the formatting rounds is, with minor modifications, machine compilable once the appropriate preamble has been prepended, but is still far from a completed ebook. The remaining work on a project, generically termed "post-processing" and typically comprising about 25–35% of the total production time, is performed off-line.

2.2 Coding for longevity

Data formats are a troublesome fact of life for long-term electronic encoding and storage of information. Electronic documents become useless when there is no easy, reliable way to recover the textual and presentational information stored in a file format.

Storage in an open, non-proprietary, plain text format guards against lossage due to lack of decoding software. The textual content of a LaTeX source file will remain accessible as long as computers can read plain text in a present-day encoding. However, LaTeX markup alone does not guarantee longevity; far from it. Used as a WYSIWYG tool, even the most capable markup language cannot capture more than a book's visual appearance.

For longevity, flexibility, and ease of maintenance, a source file needs to separate four interrelated but distinct aspects: (i) textual content (maintaining readability by both people and machines), (ii) semantic structure, (iii) visual presentation and layout, and (iv) implementation in terms of typesetting software.

Carefully-planned macros meet all four requirements, embodying these multiple layers of structure, both clarifying the code and simplifying the task of future maintainers who wish to convert today's LaTeX files into source files suitable for the typesetting software of 2050 and beyond. Technical details of DP's current practices are surveyed in Section 4 below.

3 The structure of DP

Since the production of mathematical ebooks at DP takes place within an infrastructure designed primarily for HTML-formatted projects, it is worth describing the general organization and operation of DP in parallel with the special requirements and practices of the LaTeX community.

DP is primarily an English-language site. For LaTeX projects, English-language books are generally preferred, though a number of books in French and German have also been produced. The site code currently restricts source files to the Latin-1 (`iso 8859-1`) encoding, so a book's language must be representable in Latin-1. (DP-Canada and DP-Europe can handle `utf-8` with some limitations.)

There are four major phases of ebook production at DP: content providing, proofing, formatting, and post-processing. Each has its own time commitments, skill set, and access requirements [2].

3.1 Content providing

A content provider (CP) conveys a collection of page scans, possibly including OCR output, to an experienced DP volunteer known as a "project manager". Scans may be "harvested" from a third party such as the Internet Archive, or may be scanned by the CP. A "copyright clearance" must be obtained from Project Gutenberg before scans are uploaded to DP [4].

If you would like to have a specific work transcribed at DP, please contact the author of this article or post in the "LaTeX Typesetters Team" in the DP forums.

Selecting suitable books

Books should normally be selected primarily for expected popularity or value as scholarly references. A new LaTeX project should not be initiated unless a volunteer expresses the *commitment* to post-process.

Given the current size of the LaTeX community at DP, the best books are in the vicinity of 250 pages or fewer, and contain mostly uniform, straightforward typography, and only mathematics that can be easily typeset using the AMS math environments.

Books should generally be avoided if they contain extensive typography that is relatively difficult to render in LaTeX, such as long division, tabular data with many multi-row or multi-column alignments, multi-column lists of exercises and answers, typography that changes at each paragraph (as in a geometry textbook), or large numbers of illustrations, particularly inset diagrams.

3.2 Proofing

The "distributed" portion of ebook production at DP has well-developed guidelines designed to allow most pages of most books to be processed uniformly. When questions arise of how to handle unusual constructs, volunteers may communicate with each other and with the project manager via phpBB bulletin boards. Each project has a dedicated discussion thread. There are also dozens of forums for general questions.

Normally, each page of a book passes through three rounds of proofing, named P1–P3, with successive passes made by volunteers having greater experience and ability at catching errors. Once all pages of a project have completed a round, the project is made available in the next round. At any given time, a project is "in" a specific round, and each page of a project is proofed the same number of times.

In the proofing rounds, volunteers ensure that the characters in the text file match the characters in the page scan. In other words, the focus is on content.

In a LaTeX project, the first proofing round typically involves considerable "type-in", or manual entry of characters, because OCR handles mathematics so poorly. A single page may require 10 or 15 minutes' work, a substantial fraction of the expected total preparation time.

3.3 Formatting

After the proofing rounds, each page goes through two rounds of formatting, F1 and F2. The formatting rounds capture the book's structure: chapter and section headings, quotations, footnotes and sidenotes, tables, and figures. In LaTeX projects, mathematics is coded primarily by the formatters.

For a LaTeX project, F1 entails a similar amount of type-in to P1. Additionally, a "formatting coordinator" (see Section 4) provides a "working preamble" for the project. Volunteers are expected to test-compile each page before marking it as "done", and to check the compiled code visually against the page scan. This amount of work makes F1 the most time-consuming round for LaTeX, about 10–20 minutes' work per page.

3.4 Post-processing

After a project leaves the rounds, the distributed phase is complete. The remaining work is done by a volunteer playing the role of "post-processor" (PPer).

A PPer downloads the formatted concatenated text and polishes it into an ebook, regularizing and finalizing the LaTeX code. Normally, a PPer becomes involved with a project before the project reaches the formatting rounds and serves as the formatting coordinator, ensuring the project is formatted according to the PPer's wishes.

PPing is complex and time-consuming, requiring fairly extensive planning and about 10–20 minutes' work per page for a modestly-complex book. At the same time, PPing provides an outlet for organizational creativity and typographical artistry, and is therefore one of the most satisfying and intellectually challenging tasks at DP.

3.5 Access requirements

Access to various activities at DP is granted according to time on site, number of pages processed, and/or peer review of one's work. Each DP site has its own set of certification requirements. Criteria for the DP site at `www.pgdp.net` are described here.

New volunteers are immediately granted access to P1. Access to P2 is granted once a volunteer has been registered for 21 days and has proofed at least 300 pages. Certification to work in the third round of proofing is granted by application only, upon satisfactory performance under detailed human evaluation of the volunteer's proofing work. In order to apply for P3, a volunteer must have been registered at DP for at least six weeks, and have proofed at least 150 pages in P2, and formatted at least 50 pages.

F1 access is granted with access to P2. F2 certification is granted by application only, after detailed human evaluation of the volunteer's formatting work. In order to apply for F2, one must have been registered at least 91 days and have formatted at least 400 pages.

Access to PP is granted *pro forma* by request after 400 pages have been formatted. New PPers must submit their completed projects for detailed inspection by an experienced "PP Verifier" (PPVer). The PPVer assigns a "grade" to the project based on its length and difficulty, and the number of errors present in the uploaded project. After completion of eight consecutive projects with sufficiently high grade, a PPer is given "direct upload" status, and may upload projects directly to PG without supervision.

Time commitments

Volunteers at DP devote as little or as much time to the site as they like. A page is the smallest unit of proofing or formatting, and for a LaTeX project typically entails 5–20 minutes' work. Many volunteers do just one page whenever they can, perhaps every week or few. Others find the work mildly but pleasantly addictive, and work an hour or more at a sitting, several times per week.

Compared to proofing and formatting, PPing involves an extended commitment of time and energy. An experienced PPer may be able to complete a 150-page book in as little as 40 hours, but longer or more complex books can easily absorb upward of 100 hours.

Documentation and LaTeX requirements

The guidelines for proofing, formatting, and post-processing LaTeX are detailed in a set of manuals [5]. These and other LaTeX-related information applicable to DP may be found in the DP wiki [3].

4 DP LaTeX coding strategies

This section discusses, in some technical detail, current practices for coding LaTeX at DP. Most of these ideas are not new, but neither do they seem widely-articulated. These strategies need not be studied except by volunteers who intend to post-process, but their rationale must be consciously and continually remembered when working at DP, where the page-at-a-time interface naturally leads a formatter to focus detrimentally on small-scale document structure.

4.1 Textual content

When a scanned page is OCRed, the output text contains the same line breaks as the printed book. Of course, the original pagination and line breaks need not and cannot be retained in a compiled PDF file. To the extent possible, however, line and page breaks *are* retained in the LaTeX source file. At DP, hyphenated words are rejoined, but otherwise there is no rewrapping of lines. Page separators are retained as LaTeX comments. The source file is therefore a reasonable visual copy of the original book, facilitating the tasks of proofreaders and eventual document maintainers.

Page and footnote numbers depend upon the document's pagination, and are not retained in the compiled output file. Other than this, textual content is retained in the document body. Particularly, LaTeX's auto-numbering is suppressed. Chapters, sections, numbered items, theorems, and equations are tagged manually, usually with the original numbering or labels represented as macro arguments. These labels have been assigned in the print version, and are *de facto* part of the original text.

Structural macros, e.g. `\Chapter`, `\Section`, `\Figure`, `\begin{Theorem}` and `\end{Theorem}`, or `\Proof`, normally generate text similar to the macro name, and do not generate more text than necessary. For example, even if most proofs begin with the phrase: "**Proof**: We must show that...", a `\Proof` macro would generate the word "Proof" in boldface, but would *not* generate the phrase "We must show that". The aim of LaTeX coding at DP is to separate content and typographical presentation in the document body and preamble, respectively. To the extent possible, the source file should be searchable for words and phrases appearing in the original book. Detailed knowledge of the preamble should not be prerequisite to reading the textual content of the book from the document body.

4.2 Semantic structure

A document body should contain few commands explicitly specifying how a piece of text is to be typeset. Instead, the body contains mostly mnemonic, high-level structural information: "this is a chapter", "this is a theorem", "this is a figure", and so forth.

The goal of semantic coding frequently turns out to be non-trivial to implement. Proofers and formatters see only one page of a book at a time. How, without inspecting a large fraction of pages, is a formatter to know the meaning of a boldface, run-in heading, or of centered italics? What if only some theorem statements are italicized; are the italics significant, or was the typesetter merely inconsistent?

At DP, a "formatting coordinator" inspects the entire book before the project leaves the proofing rounds, notes the major semantic structures and any typographical irregularities, then writes a "working preamble" for use during the formatting rounds. Ideally, the working preamble macros satisfy a number of disparate requirements. They are easy to remember, do not require formatters to type much, give a good approximation to the page scan when a formatter test-compiles a single page, and capture enough information to match the book's global typography (running heads, table of contents entries, PDF bookmarks, hyperlinks, and the like) in post-processing. For example, the text of a chapter heading might progress through the proofing and formatting rounds like this:

```
CHAPTER III: Curvature   % Proofed
\CHAPTER{III: Curvature} % Formatted
\Chapter{III}{Curvature} % Uploaded
```

All the typographical work is centralized in macro definitions.

As suggested by this code snippet, structural macros in the working preamble should *not* normally be standard LaTeX commands such as `\chapter`. Sectioning commands of LaTeX's document classes are designed with different aims than than are required at DP: They provide unwanted numbering, and are often non-trivially integrated into the document class using modern typographical assumptions. In a DP-era book, for example, a new chapter might not re-set the running head, might not start recto, and might not even begin on a new page. However, redefining the `\chapter` command accordingly also changes the behavior of the table of contents, preface, appendices, and index, probably in undesired ways.

Instead, it's preferable to add an interface layer between structural macros in the body and their implementation in terms of LaTeX commands. A `\Chapter` command in the working preamble might be implemented with the LaTeX `\section*` command. In post-processing, only the macro definition, *not the formatters' code*, needs to be modified in order to achieve the necessary typographical and cross-referencing effects.

This technique beneficially centralizes the document's dependence on the compilation engine. If typesetting software changes, only the macro definitions need modification, not every occurrence in the document body. Amplifications of this strategy are used at DP to help ensure stylistic uniformity, and to match the original typography with relative ease.

4.3 Visual presentation

DP volunteers express a wide range of opinions on how much effort should be spent making a book resemble the original, or whether ebooks should be optimized for printing (two-sided layout, generous margins) or for ebook readers (single-sided layout, very tight margins, colored hyperlinks).

There is an obvious trade-off between attractive layout on one hand and flexibility in accommodating different ebook readers on the other. This trade-off is strongly dependent on the original book; floating tables and illustrations, or even complex mathematical displays, are difficult to lay out well unless the text block size is known. As ebook readers with varying screen sizes proliferate, post-processors will encounter increasing difficulty in ensuring that finished ebooks look good on a variety of hardware.

Centralized structural coding described above facilitates the task of creating a flexible, camera-quality ebook.

Structural coding also sidesteps an issue that plagues WYSIWYG authors: Ensuring visual consistency. If section headings are printed in centered boldface type and these typographical attributes are specified explicitly for each section, the section headings are all but impossible to make identical, or to tweak and maintain.

These facts of document design are easy to see at the level of authoring an entire document, but are remarkably easy to forget when one is working one page at a time in the DP formatting rounds. The experience of years past shows that even experienced LaTeX coders incline toward hard-coding visual markup under real-life circumstances.

4.4 Implementation

In addition to the previously-noted benefits of separating structure, presentation, and content, well-planned semantic coding and encapsulating interfaces can guard against changes to external software.

A LaTeX source file obviously depends for compilability on external packages and the LaTeX kernel itself. For the LaTeX kernel and "core" packages, the need for backward compatibility helps ensure that *user interfaces* do not change. By contrast, kernel and package *internals* are all but guaranteed to be re-written beyond recognition on a time scale of decades.

On occasion in years past, LaTeX-knowledgeable post-processors at DP have concluded that a book's typography can be matched elegantly by redefining macros in a standard document class. In retrospect, this strategy is ill-advised: It relies on software internals over which the post-processor has no control.

At DP, the goals of structural markup and consistent visual presentation are achieved through factoring of typographical "responsibilities". A three-level scheme, inspired by object-oriented programming, has proven itself over dozens of projects.

Structural macros At the highest level, used in the document body, are purely structural macros needed to mark the book's semantics: `\Chapter`, `\Section`, `\Proof`, and the like.

Semantic operations In even a moderately complicated book, multiple sectioning commands need to perform identical abstract typographical operations, such as "set the running heads", "write an entry to the table of contents", "create a PDF bookmark", "include a graphic with a default width from a specified directory", or "get to a recto page, clearing the stale running head on the preceding verso page if necessary". For flexibility, visual consistency, and ease of maintenance, these operations should be factored out. Macros at this second level are not normally called directly in the document body, but only in the preamble, in the definitions of structural macros.

Depending on the book's complexity, common features among *semantic* macros may be best factored out as well. Generally, therefore, even second-level macros might *not* be implemented directly in terms of LaTeX commands.

Visual implementation The commands used to effect the visual presentation lie at the third level. These include both abstract operations such as "set the format and location of the page numbers" or "select the font of the running heads", and specific, concrete operations such as "make this text boldface". These macros, at last, are implemented in terms of standard LaTeX commands, including facilities provided by external packages.

4.5 Remarks and caveats

Abstraction and encapsulation do not always buy flexibility, and should not be used needlessly. Standard LaTeX macros, such as mathematical symbols, AMS displayed math environments, and `\footnote` commands are used routinely.

Naturally, a macro system must be designed from the top downward, based on inspection of the entire book. First determine the necessary semantic structures, then find and factor out typographical and cross-referencing operations common to two or more structural operations, and finally implement any common operations in terms of LaTeX commands.

The three layers of abstraction above are important mostly when one wishes to mimic the printed appearance of the original book. When a project warrants this level of coding, the typographical appearance can be fine-tuned easily, accurately, and consistently.

For simpler projects, this scheme may be overly elaborate. Further, if the default appearance of a standard document class is acceptable, coding semantically in terms of LaTeX's sectioning macros may be entirely workable.

Using primarily structural macros in the document body helps ensure the book will be machine-convertible to other formats, even formats not yet in existence, with as little fuss as possible. No one holds the illusion that DP's LaTeX projects can be trivially converted to other formats. However, a thoughtfully-coded ebook should be convertible to a new format with perhaps a few hours' work, compared to the dozens or hundreds of hours required to digitize the project initially.

Floats and inset illustrations

Figures, tables, and complex displayed mathematics are simply a problem for current ebook readers, whose screens may be only a few inches wide.

Inset illustrations are a common cause of "brittle" documents, code whose compiled quality depends sharply on the size of the text block. The `wrapfig` package is powerful, but has relatively tight constraints on how it can place diagrams. In particular, a single paragraph cannot contain more than one `wrapfigure` environment, and mid-paragraph placement requires manual coding.

It is currently considered acceptable at DP to hard-code the placement of wrapped illustrations, but arguably it is more robust (though less pleasant typographically) to use ordinary `figure` environments instead.

DP-specific coding tricks

Proofers and formatters at DP commonly make in-line notes regarding misspellings, visual obscurities, notational inconsistencies, even factual errors. Two simple macros, `\DPnote` and `\DPtypo`, are used to leave notes in the source file. `\DPnote` is a one-argument null macro. `\DPtypo` accepts two arguments, the original text and the putative correction. Changes are trivially switched on (or off) by changing one line of preamble code. Future maintainers can easily review all such changes by searching the source file for the macro name.

DP post-processors commonly use the `ifthen` package and a boolean switch to control layout suitable for printing or for an ebook reader. Again, the behavior is trivially toggled by editing one line in the source file. The scope of this technique is limited, however. Unless a book contains few or no inset diagrams, the respective print and screen layouts must in practice have the same text block size.

5 The future

This is potentially an exciting era for LaTeX at DP; training guidelines have been written and a stable work flow has emerged after an early period that relied on the skills of specific individuals. Whether or not DP contributes substantially to the preservation of mathematical literature in the coming years depends on its ability to build a self-sustaining community of dedicated volunteers.

Future projects should be chosen according to criteria ranging from scholarly or pedagogical value to expected popularity. Content providers must candidly evaluate a book's "value" and typographical needs, and appraise whether or not the book justifies the necessary labor to produce in LaTeX.

LaTeX-capable formatters are needed simply to distribute large amounts of work among many volunteers. What takes one formatter a month can be done by ten volunteers in a few days. Encouraging students to work at DP can both provide valuable LaTeX coding practice and serve as an introduction to document design and planning.

For students writing a thesis, post-processing can be an avenue to working with book-length manuscripts. Naturally, PPing at DP has distinctive requirements from "ordinary" mathematical authorship, but many skills are transferable.

The contribution of just one proofed or formatted page per day from a dozen new volunteers would substantially increase DP's current LaTeX throughput. Thoughtful suggestions for new content will help ensure that important mathematical works will be available electronically for posterity.

References

[1] The catalog of LaTeX projects produced at DP, `http://www.pgdp.net/wiki/List_of_LaTeX_projects/Posted`.

[2] The DP wiki, `http://www.pgdp.net/wiki/Main_Page`.

[3] The DP wiki LaTeX links, `http://www.pgdp.net/wiki/LaTeX_Links`.

[4] The Project Gutenberg copyright clearance page, `http://www.gutenberg.org/wiki/Gutenberg:Copyright_How-To`.

[5] LaTeX documentation for DP, `http://mathcs.holycross.edu/~ahwang/pgdp/dptest/index.html`.

Andrew Hwang