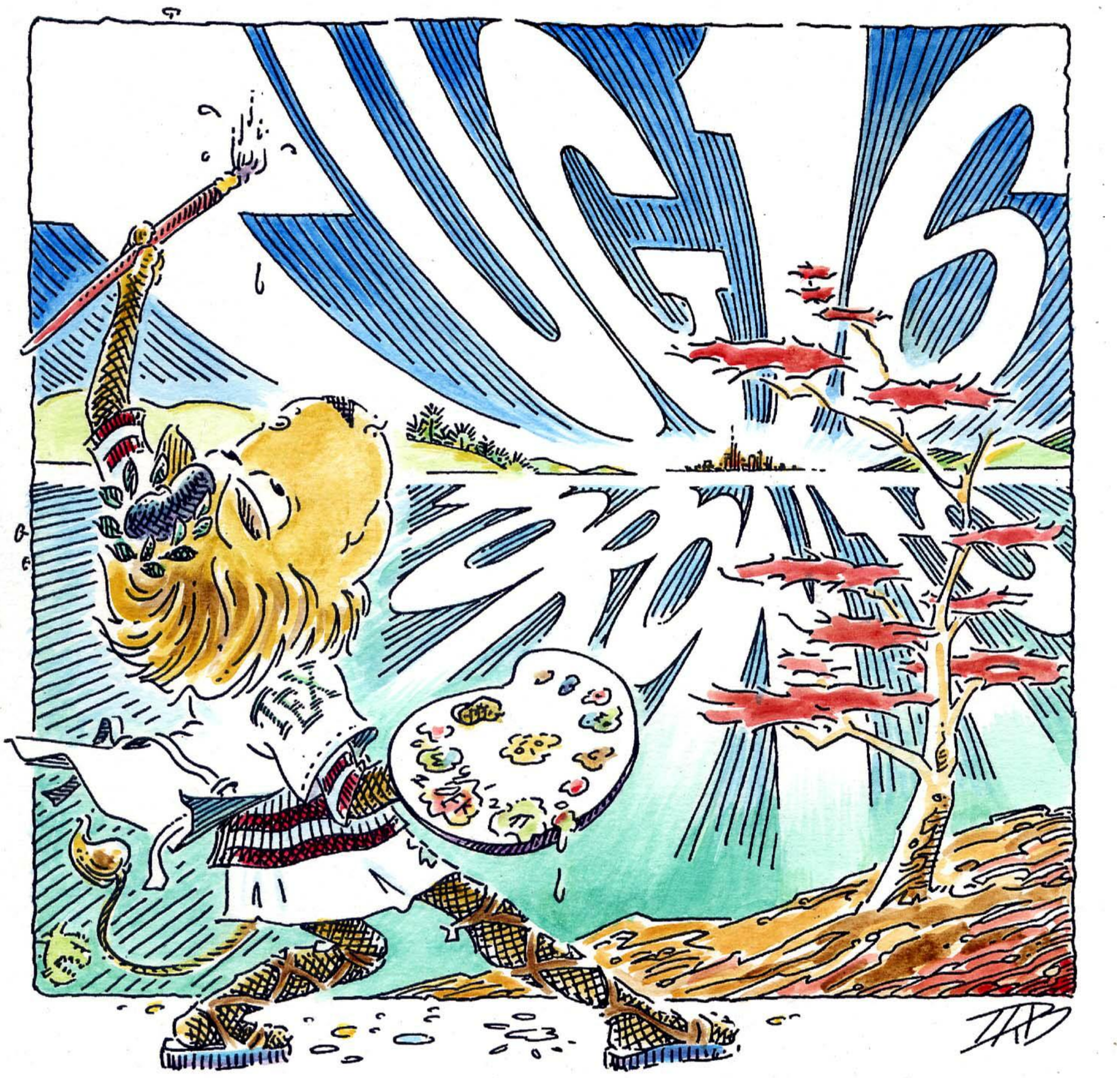


TUG 2016

Workshops and Presentations on LaTeX, TeX, PDF, Unicode, and more

Bond Place Hotel — Toronto, Canada

July 25–27, 2016

Sponsors

TEX Users Group ▪ **DANTE e.V.** The Porcupine's Quill ▪
River Valley Technologies — UK ▪
with special assistance from individual contributors. *Thanks to all!*

Special guests

Charles Bigelow, Bigelow & Holmes ▪ Robert Bringhurst, Quadra Island, BC ▪ Kevin Larson, Microsoft

Conference committee

Pavneet Arora ▪ Karl Berry ▪ Robin Laakso ▪ Jim Hefferon ▪ Steve Peter

Participants

Pavneet Arora, Bolton, ON
Amartyo Banerjee, TNQ, India
Abdelouahad Bayar, Cadi Ayyad University, Morocco
Kaveh Bazargan, River Valley Technologies, UK
Nelson Beebe, University of Utah
Barbara Beeton, AMS
Karl Berry, Bandon, OR
Johannes Braams, Zoetermeer, The Netherlands
Jozo Capkun, Caledon, ON
David Casperson, Univ. of Northern British Columbia
Jaeyoung Choi, Seoul, Korea
Joe Clark, Toronto, ON
Jennifer Claudio, Oak Grove High School
Sue DeMeritt, Center for Communications Research, La Jolla, CA
Mercedes Dollard, Pittsburgh, PA
Michael Doob, University of Manitoba
Yukitoshi Fujimura, Ichikawa-shi, Japan
Christian Gagné, Université Laval
Federico Garcia-De Castro, Alia Musica Pittsburgh
Peter Giunta, MIT
Steve Grathwohl, Duke University Press
Katie Harding, Dartmouth College
Jim Hefferon, Saint Michael's College
Tim Inkster, The Porcupine's Quill
John Eddie Kerr, Wellington Law Association Library
Stefan Kottwitz, Lufthansa Industry Solutions
Yusuke Kuroki, Yokohama, Japan
Robin Laakso, TEX Users Group
Richard Leigh, St Albans, UK
Lothar Meyer-Lerbs, Bremen, Germany
Frank Mittelbach, LATEX3 Project
T Rishikesan Nair, River Valley Technologies, India
Kim Nesbitt, Canadian Journal of Economics
Steve Peter, TUG
John Plaice, Montreal, Canada
Cheryl Ponchin, Center for Communications Research, Princeton, NJ
Geoffrey Poore, Union University, Tennessee
Norbert Preining, Accelia
C V Rajagopal, River Valley Technologies, India
Arthur Reutenauer, Royal Opera House
Volker RW Schaa, DANTE e.V.
Herbert Schulz, Naperville, IL
Heidi Sestrich, Carnegie Mellon University
Michael Sharpe, UC San Diego
A M Shanmugam Pillai, River Valley Technologies, India
Keiichiro Shikano, Tokyo, Japan
Matthew Skala, IT University of Copenhagen
Michael Sofka, Rensselaer Polytechnic Institute
Christina Thiele, Nepean, ON
David Tulett, Memorial University, Newfoundland
Boris Veytsman, George Mason University
David Walden, East Sandwich, MA

TUG 2016 program

Monday July 25	8:00 am	*registration*	
	8:45 am	Pavneet Arora, Bolton, ON	*Opening: Passport to the TeX canvas*
	9:30 am	Geoffrey Poore, Union Univ.	*Advances in PythonTeX*
	10:00 am	Stefan Kottwitz, Lufthansa Industry Sol.	*TeX in industry I — programming Cisco switches using TeX*
	10:30 am	*break*	
	10:45 am	Stefan Kottwitz	*TeX in industry II — designing converged network solutions*
	11:15 am	Amartyo Banerjee, S.K. Venkatesan, TNQ	*A Telegram bot for printing LaTeX files*
	11:45 am	Frank Mittelbach, LaTeX3	*Alice goes floating — global optimized pagination including picture placements*
	12:45 pm	*lunch*	
	1:45 pm	Michael Doob, Univ. of Manitoba	*baseball rules summary*
	2:00 pm	Boris Veytsman, George Mason Univ.	*Making ACM LaTeX styles*
	2:30 pm	Norbert Preining, Ishikawa, Japan	*Security improvements in the TeX Live Manager and installer*
	3:00 pm	Arthur Reutenauer, Royal Opera House	*The TeX Live M sub-project*
	3:45 pm	*break*	
	4:00 pm	Kevin Larson, Microsoft	*Reading between the lines: Improving comprehension for students*
	≈ 5 pm	*end*	
Tuesday July 26	8:25 am	*announcements*	
	8:30 pm	Kaveh Bazargan, River Valley Technologies, UK	*A graphical user interface for TikZ*
	9:00 am	Matthew Skala, IT Univ. of Copenhagen	*Astrological charts with* `horoscop` *and* `starfont`
	9:30 am	David Tulett, Memorial Univ.	*Development of an e-textbook using LaTeX and PStricks*
	10:00 am	Christian Gagné, Univ. Laval	*An Emacs-based writing workflow inspired by TeX and WEB, targeting the Web*
	10:30 am	*break*	
	10:45 am	Arthur Reutenauer, Mojca Miklavec	*Hyphenation past and future:* `hyph-utf8` *and* `patgen`
	11:15 am	Jim Hefferon, Saint Michael's College	*A LaTeX reference manual*
	11:45 am	Federico Garcia-De Castro, Alia Musica	*TeXcel?*
	12:15 pm	Jennifer Claudio, Oak Grove High School	*A brief reflection on TeX and end-user needs*
	12:45 pm	*lunch*	with typeforming video (40 min)
	2:00 pm	Jaeyoung Choi, Seoul, Korea	*MFCONFIG: Metafont plug-in for the Freetype rasterizer*
	2:30 pm	Michael Sharpe, UC San Diego	*New font offerings — Cochineal, Nimbus15 and LibertinusT1Math*
	3:00 pm	*break*	
	3:15 pm	Robert Bringhurst, Quadra Island, BC	*The evolution of the Palatino tribe*
	4:15 pm	*TUG meeting*	
	≈ 5:15 pm	*end*	
	5:15 pm	Herbert Schulz, Naperville, IL	*Workshop: TeXShop tips & tricks*
Wednesday July 27	8:25 am	*announcements*	
	8:30 am	Abdelouahad Bayar, Cadi Ayyad Univ.	*Towards an operational (La)TeX package supporting optical scaling of dynamic mathematical symbols*
	9:00 am	John Plaice, Montreal, QC	*Zebrackets: A score of years and delimiters*
	9:30 am	Charles Bigelow, Bigelow & Holmes	*Probably approximately not quite correct: Revise, repeat*
	10:30 am	*break*	
	10:45 am	David Walden, East Sandwich, MA	*Some notes on the history of digital typography*
	11:15 am	Tim Inkster, The Porcupine's Quill	*The beginning of my career*
	12:15 pm	*lunch*	with road painting video (10 min)
	1:45 pm	*group photo*	
	2:00 pm	Joe Clark, Toronto, ON	*Type and tiles on the TTC*
	3:00 pm	*break*	
	3:15 pm	Jennifer Claudio	*The case for justified text*
	3:45 pm	Boris Veytsman	*Are justification and hyphenation good or bad for the reader?*
	4:15 pm	Charles Bigelow	*Looking for legibility*
	≈ 5:15 pm	*end*	
	6 pm	*Type and Tile Subway Tour, Joe Clark*	typography discussion at 3–5 subway stops.

Excursion outline

See `http://tug.org/tug2016/excursions.html` for detailed schedule.
All return times are necessarily approximate.

Monday July 25 — baseball

5:35 pm	baseball excursion walkers depart Bond Place
7:35 pm	game time at Rogers Centre

Tuesday July 26 — bookshop, dinner

6 pm	Eliot's Bookshop browsing
7:30 pm	informal dinner, Kaiju Toronto

Wednesday July 27 — subway typography, dinner

6 pm	Type and Tile Tour, Joe Clark — typography discussion at 3–5 subway stops.
7 pm	informal dinner, Jazz Bistro
8 pm	music by Sue Foley, Jazz Bistro

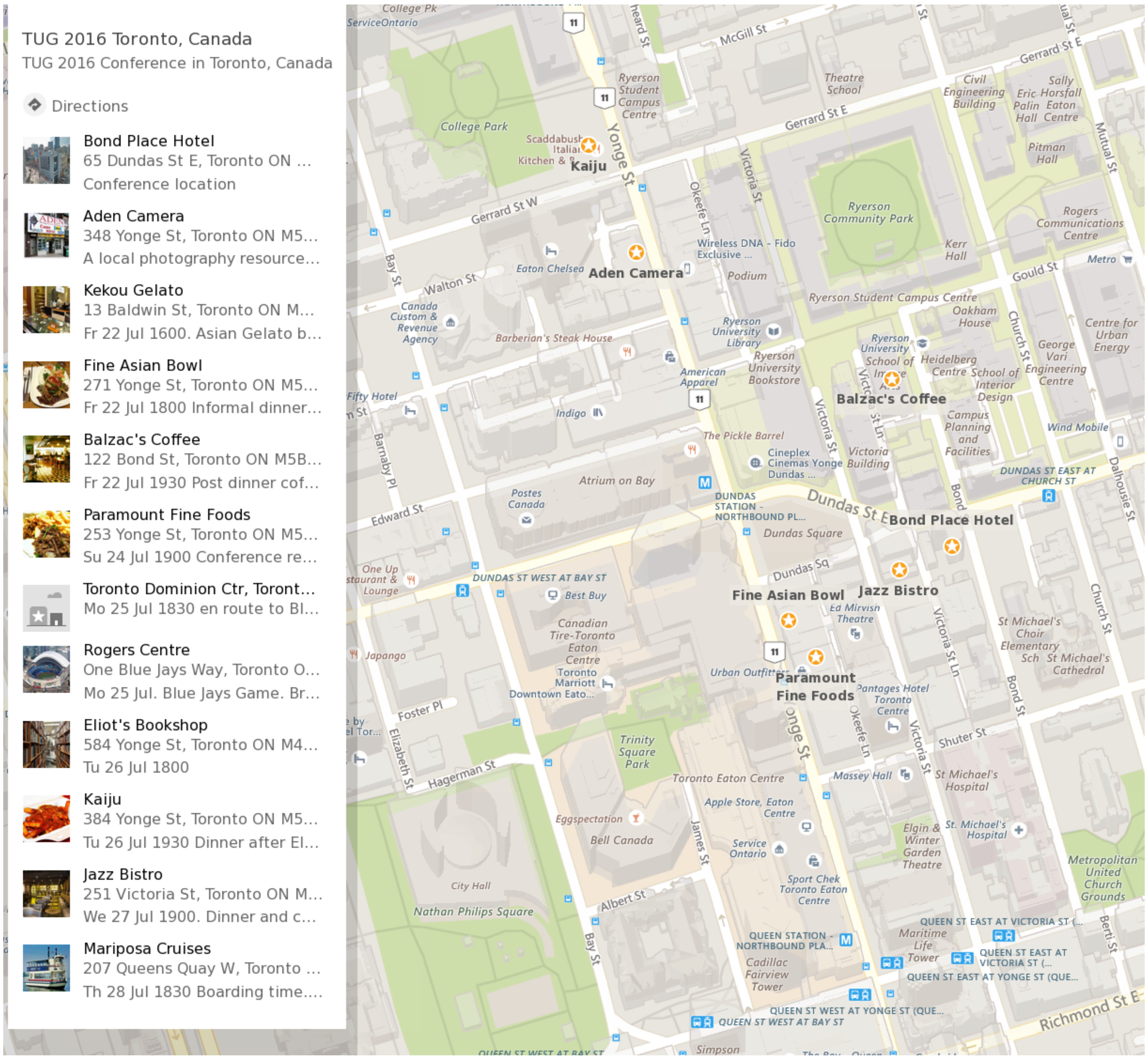
Thursday July 28 — typography, banquet

8:30 am	depart Bond Place for typography day
5 pm	arrive back at hotel
6 pm	depart Bond Place for harbor cruise and banquet
10 pm	arrive back at hotel

Friday July 29 — Georgian Bay

9 am	depart Bond Place
6 pm	arrive back at hotel

Hotel vicinity.



TUG 2016 Toronto, Canada
TUG 2016 Conference in Toronto, Canada

Directions

Bond Place Hotel
65 Dundas St E, Toronto ON …
Conference location

Aden Camera
348 Yonge St, Toronto ON M5…
A local photography resource…

Kekou Gelato
13 Baldwin St, Toronto ON M…
Fr 22 Jul 1600. Asian Gelato b…

Fine Asian Bowl
271 Yonge St, Toronto ON M5…
Fr 22 Jul 1800 Informal dinner…

Balzac's Coffee
122 Bond St, Toronto ON M5B…
Fr 22 Jul 1930 Post dinner cof…

Paramount Fine Foods
253 Yonge St, Toronto ON M5…
Su 24 Jul 1900 Conference re…

Toronto Dominion Ctr, Toront…
Mo 25 Jul 1830 en route to Bl…

Rogers Centre
One Blue Jays Way, Toronto O…
Mo 25 Jul. Blue Jays Game. Br…

Eliot's Bookshop
584 Yonge St, Toronto ON M4…
Tu 26 Jul 1800

Kaiju
384 Yonge St, Toronto ON M5…
Tu 26 Jul 1930 Dinner after El…

Jazz Bistro
251 Victoria St, Toronto ON M…
We 27 Jul 1900. Dinner and c…

Mariposa Cruises
207 Queens Quay W, Toronto …
Th 28 Jul 1830 Boarding time.…

Leyla Akhmadeeva, Rinat Gizatullin, Boris Veytsman

Are justification and hyphenation good or bad for the reader?

Since the early days of typography, text justification was considered a necessary feature of well-typeset copy. The type block should be rectangular and grey. Even the "New Typography" by Jan Tschichold, while rejecting many dogmas of classical typesetting, still retained text justification and, as a consequence, hyphenated text.

In this work we study the impact of justification and hyphenation on the reading speed and comprehension. The participants in the experiment read justified and non-justified (and unhyphenated) texts and answered simple questions about them.

We found the difference between the justified and unjustified texts very small. Still it is present:

1. Justified texts were read slightly faster.
2. Unjustified texts gave slightly better results in the recall tests. The probability of the correct answer for a multiple choice question increased from 0.89 to 0.93: this might mean four extra points on a 100-question test.

Amartyo Banerjee, S.K. Venkatesan

A Telegram bot for printing LaTeX files

A proof of concept of a Telegram bot running on a Raspberry Pi is described here. The bot accepts a LaTeX file from the user, process it and send back to the user a PDF file resulting from that processing.

In the conference paper the following things are discussed:

1. The genesis of the idea of the bot.
2. The use case for the bot as it exists at present, and additional functionality implemented.
3. The learning process and obstacles encountered in development.
4. Additional functionality planned to be implemented, such as processing a multi-file LaTeX document, and printing the PDF file.
5. Steps needed to make it production-ready, including robust error handling and proper input sanitization.
6. Potential for a complete rewrite to meet scalability requirements and get around file download limitations in the Telegram bot API.

Abdelouahad Bayar

Towards an operational (LA)TeX package supporting optical scaling of dynamic mathematical symbols

In processing of digital documents containing mathematical formulas, the handling of dynamic mathematical symbols is still a significant and difficult problem. In fact, a tool to compose mathematics must support the typing of variable-sized symbols taking care of optical scaling and reaching the quality of metal typesetting. Until now, no tool provides these possibilities in a direct and operational way. This contribution highlights and puts into practice the basic steps to develop a (LA)TeX package directly based on a parameterized Type 3 PostScript font. This package will provide to (LA)TeX end-users a tool to compute in the usual way mathematical formulas consisting of dynamic mathematical symbols and taking into account optical scaling

Formatting (LA)TeX documents using this package is achieved without requiring special environments or external programs. The concept of using parameterized Type 3 fonts directly with (LA)TeX commands can yield an accurate and straightforward way to manage dynamic graphics in documents formatted under (LA)TeX, including logo graphics, for example.

Keywords: (LA)TeX, PostScript Type 3, dynamic mathematical symbols, optical scaling.

Kaveh Bazargan

A graphical user interface for TikZ

Ti*k*Z is a powerful vector graphics program. The capabilities are far higher than those of any interactive graphics program, e.g. Adobe Illustrator. The fact that Ti*k*Z can be coded logically, keeping the structure of a graphic intact has many advantages, including the generation of "semantic" SVG output, allowing for better accessibility of graphics, e.g. for blind and visually impaired readers. The main barrier against popular usage of Ti*k*Z outside the TeX community is that it is purely code driven, and has a steep learning curve.

At River Valley we have a long term program to produce rich, semantic illustrations that can be accessible to readers who are blind or visually impaired. We have chosen Ti*k*Z as the engine to generate these illustrations.

To make easier to create the diagrams in the first place, we have been working on a graphical user interface (GUI) to generate Ti*k*Z code and to show its output in near real-time. The first implementation addresses the relabelling existing illustrations (vector or bitmap) using Ti*k*Z. The system has the following advantages:

- Relabelling can be done as fast or faster than conventional methods
- Labels do not change size as figures are resized (similarly to Pinlabel)
- Labels are saved as text and can be spell-checked along with the main text of a TeX file

I will give a live demo of the software.

Charles Bigelow

Looking for legibility

Scientific studies of legibility and their contributions to typography and type design from the 19th to the 21st century. With amusing anecdotes, ingenious contraptions, clueless assumptions, cooked results, Herculean efforts, and occasional progress toward understanding one of the most puzzling cultural activities.

Charles Bigelow

Probably approximately not quite correct: Revise, repeat

The evolution of Lucida Math fonts over 30 years of intermittent convergence toward a semi-solution by successive aesthetic and graphic-semantic approximations, including stumbles. Includes "oh-oh, zero" and super-ellipsism.

Robert Bringhurst
The evolution of the Palatino tribe

Palatino is not just a single typeface but a large and varied group of faces: a taxonomic tribe produced over more than half a century. The members include Aldus, Enge Aldus, Aldus Nova, Heraklit, Michelangelo, Phidias, Sistina, and Zapf Renaissance, as well as foundry Palatino, Linotype Palatino, American Export Palatino, Linofilm Palatino, PostScript Palatino, Palatino Nova, and Palatino Sans.

This constellation of type designs was Hermann Zapf's most ambitious and enduring design project. It began with the "Medici" sketches of 1948 — which led to the first trial cutting of foundry Palatino roman by August Rosenberger at the Stempel Foundry, Frankfurt, in 1949 — and it continued through 2006, when the last authentic members of the group were drawn on screen under Zapf's direction by Akira Kobayashi in Bad Homburg. In between these dates, the underlying designs adapted again and again to changing conditions, represented by the Linotype machine, Linofilm and other phototype machines, and a variety of pre-Postscript digital systems.

Zapf was not the only type designer whose career spanned the tumultuous transitions from metal type to phototype to digital type, but Palatino and its relatives appear to be unique in the complexity of their evolution and the multiplicity of their successive adaptations, under the hand of the original designer, to repeatedly changing methods of typesetting and printing.

Robert Bringhurst has argued for many years that the most promising system of typeface classification is based on botanical and zoological taxonomies. His new book, *Palatino: The Natural History of a Typeface*, to be published this summer by the Book Club of California, is an extended test of this thesis. Over many years of research, he has also accumulated hundreds of illustrations documenting the artistry and care, and the industrial advances and collapses, involved in creating these components of our typographic heritage.

Jennifer Claudio
The case for justified text

Documents must be aesthetically pleasing without appearing deliberately designed. One of the basic functions built into most word processing software is text justification. The algorithms behind this function, however, vary based on the software and the preset rules within the software. Some criticisms include compromised readability. Defenders of justified text argue that as long as the typeface is appropriately sized and kerned, justification does not detract from readability. This presentation succinctly demonstrates the behavioral differences visible in WordPerfect, Word, InDesign, and LaTeX, and examines the ability for people who read common printed media to notice the differences.

Jennifer Claudio
A brief reflection on TeX and end-user needs

TeX attracts users who seek a robust method of creating precise typographic products. However, beginning with the instinct to disambiguate the pronunciation of TeX and LaTeX, new users often feel daunted by the so-called learning curve of TeX and its relatives. Given time to learn the syntax and experience in troubleshooting errors, many fare well; however, a population exists that would benefit from the use of TeX who have insufficient time or comfort in the field.

This presentation describes the following: 1) personal use of products that have been shared by other members of this TeX user group community; 2) the quest to recruit new users, abusers, and developers into the TeX community; and 3) requests for specific end-user products.

Michael Cohen, Blanca Mancilla, John Plaice
Zebrackets: A score of years and delimiters

In this paper, we present the resurrection of the *Zebrackets* project, originally initiated by the first author 20 years ago, with which parentheses and brackets are *zebra-striped* with context information. There are two reasons for this innovation: first, to improve visual presentation of the necessary linearization of hierarchical structures in text, and second, to make a first step away from the assumption that documents must be built up from a set of unchanging atoms called characters.

Christian Gagné
An Emacs-based writing workflow inspired by TeX and WEB, targeting the Web

When using Emacs, there are at least four different things called 'macros' — and I will talk about all four. There are the Emacs Lisp macros in which Emacs is written, there are keyboard macros recorded in a different command language which is not Lisp, there are Org-mode lexical macros and finally there are the TeX macros that one writes and uses in the AUC-TeX mode.

My presentation will focus on Org-mode macros, because they are of special interest to TeX users who wish to take advantage of their skills when the publication target is the Web. These lexical macros turn out to be a decisive advantage of Org over other common lightweight markup notations such as Markdown, because they bring some of the power of writing in `WEB` to the Web.

First, I wish to present the practical method I have developed with my literary colleagues for the integration of the Markdown and XML which they produce with the LaTeX, Org and HTML5 which I produce after them. Then, I would like to talk about my notational pipe dream: a minimalistic substitution syntax, *enclosure* := *emphasis*, inspired by the substitution rules of equational logics, which would be processed by Elisp code, along with a general, human-writable style language that would be translated to both TEX typographic commands and CSS.

Federico Garcia-De Castro
TEXcel?

I recently discovered the surprising fact that TEX seems to be more appropriate for keeping financial records, and especially preparing different kinds of reports (to funders, to the IRS, by season, by project, by calendar year), than the spreadsheets that I had been using, and which had become highly convoluted as years of information piled up. This presentation is intended somewhat at the meta level: why would it be that TEX provides better for financial tracking than Excel?

Jim Hefferon
A LATEX reference manual

The unofficial LATEX Reference Manual project (`http://home.gna.org/latexrefman`) aims to provide a Free document that summarizes the features of core LATEX2ε. The idea is to be a reference manual, not an introduction and not a book. The information it contains is available from other LATEX sources and documentation, but is scattered. This work brings it together.

`latexrefman` is written in Texinfo, so output comes in a variety of forms, including HTML and PDF. The focus is on content rather than formatting. The manual has been included in free TEX distributions for many years, under the package name `latex2e-help-texinfo` (for historical reasons).

The document originated with George Greenwade, was (incompletely) updated for LATEX 2.09 by Stephen Gilmore, and (incompletely) for LATEX2ε by Torsten Martinsen. Since 2008, it has been maintained by Karl Berry, joined recently by Vincent Belaïche and, independently, me.

The manual is being actively maintained, and LATEX users should be aware of it as an available resource. In addition, I hope that people will consider contributing — a person who is knowledgable but perhaps not expert can accomplish good things here, in a step-by-step way. That includes working on translations — it is available in French (thanks to Vincent), and Spanish (thanks to a previous volunteer, Nacho Pacheco); the Spanish needs a new maintainer.

Tim Inkster
The beginning of my career

Tim Inkster was one of any number of English undergraduates at the University of Toronto in the late 1960s who were entranced by the lure of Stan Bevington's shop in the alley behind 401 Huron Street.

Inkster applied for an entry-level position at Coach House, the first time, in 1969, and then a second time a couple of years later.

Unable to secure gainful employment, Inkster felt he had little choice but to start his own small press, the Porcupine's Quill (1974), which has led to bronze and silver medals at Leipzig, a Citation from the Art Directors' Club of New York, and the Order of Canada (2009) for both Tim and his wife, Elke, "For their distinctive contributions to publishing in Canada and for their promotion of new authors, as co-founders of the Porcupine's Quill, a small press known for the award-winning beauty and quality of its books."

Sungmin Kim, Jaeyoung Choi, Geunho Jeong
MFCONFIG: Metafont plug-in for the Freetype rasterizer

One of the strengths of Metafont is to show various styles of fonts by changing values of the parameters, which represent font styles.

This advantage can be applied to letters of Roman alphabet characters and complicated CJK characters. However, Metafont can't in general be used by font engines directly. It must be changed to outline font format to be used in the current computing environment.

Also, Metafont doesn't need to create font families like Bold, Italic and Bold-Italic because it can generate various styled fonts automatically by just changing values of parameters. Therefore Metafont can reduce development time and cost required to produce a font family.

In this paper, we propose a MFCONFIG module, which enables Metafont to be directly used on GNU/Linux. It must be installed with the popular rasterizer, Freetype. MFCONFIG is a plug-in module to `fontconfig` library of the Freetype engine, which makes Freetype engine compatible with current digital font formats of both bitmap and outline fonts. It supports various styled fonts, generated from Metafont with different parameters.

Stefan Kottwitz
TEX in industry I — programming Cisco switches using TEX

Pure text-based, macro-based, TEX-programmed switch configurations which I use at my work at Lufthansa and in cruise ship projects. A brief description is at `http://tex.tips/programming-network-switches`. A demonstration of a non-standard use of TEX in industry.

Stefan Kottwitz
TEX in industry II — designing converged network solutions

Pure graphics, with efficient use of TEX and Ti*k*Z for programming drawings of network architectures (LAN, wi-fi, VoIP, ...). Efficient in the sense of my note at `http://tex.stackexchange.com/a/297029/213`. Quick sample notes: `http://tex.tips/tag/industry`. Example drawings: `http://tex.tips/LAN-1-2.pdf`.

Presenting a showcase what can be done with TEX (instead of Visio, PowerPoint or CAD) and how, with a view toward efficiency (time pressure on projects).

Kevin Larson

Reading between the lines: Improving comprehension for students

While reading is arguably a student's most important skill, the technology of reading is relatively unchanged. Can the power of computing improve a student's reading comprehension? We will discuss what has been learned about typography in the last 500 years, about reading psychology in the last 100 years, and what technology can be invented right now.

Bio: Kevin Larson works for Microsoft's Advanced Reading Technologies team. He collaborates with type designers, reading psychologists, and engineers on improving the onscreen reading experience. Kevin received his PhD for studies of reading acquisition.

Geoffrey Poore

Advances in PythonTEX

The PythonTEX package allows Python code and other programming languages to be embedded in LATEX documents. In this talk, I will provide an overview of recent developments. Code typesetting has been improved by several new features, including automatic line breaking with fine-grained control over break locations. New PythonTEX commands and environments simplify macro programming that mixes LATEX with Python or other programming languages. These make it easier to work with Python or another language without worrying about expansion, tokenization, catcode trickery, and similar LATEX fun. Adding PythonTEX support for languages beyond Python has been streamlined with a new template system. In some cases, it is now possible to capture the output of interactive command-line programs.

Norbert Preining

Security improvements in the TEX Live Manager and installer

Since the switch to the current distribution method and the introduction of network installs and updates, some years ago, many things have changed in the TEX (Live) world. But one thing has not kept up with the new distribution methods: security.

Until now, there has been almost no verification of a package as downloaded from the CTAN mirrors compared to the original package created in TL. Although we have been shipping MD5 checksums and sizes in the accompanying information, these were used only in rare instances (namely, when restarting a failed installation).

We report about the recent improvements and consistent confirmation of checksums and sizes of the downloaded packages, as well as (future?) improvements regarding strong cryptographic signatures of the package information.

Arthur Reutenauer, Mojca Miklavec

Hyphenation past and future: `hyph-utf8` *and* `patgen`

(There won't be enough time for all of this.)

1. Regenerating `patgen`:
In the course of my work about hyphenation patterns, I often wondered about overhauling the program `patgen`, that given a list of hyphenated words, produces patterns (it's the *pat*tern *gen*erator). This venerable program had been written by Frank Liang as he was working on his well-known algorithm, and had essentially not changed in over thirty years, except for an extension at the beginning at the 1990s to support 8-bit encodings, from 7 bits originally. A later rewrite in C++ proved completely unusable because it used mysterious tricks that worked only for one specific version of one specific compiler.

It served the community well, producing many sets of hyphenation patterns for different languages, but it had a number of shortcomings — lack of Unicode support to start with — and it seemed time to resume work on it, be it only to re-explore the main algorithm. Earlier this year I did just that, using Ruby which is one of my languages of choice these days. I will make a public release soon and will share some of the lessons learnt along the way, and future plans.

2. `hyph-utf8`, eight years on:
Hyphenation has always been central in TEX, to the point that Donald Knuth had one of his students, Frank Liang, develop an algorithm for it, and create a set of patterns for English. Over the years, patterns have been created for many languages and in 2008 we set to convert all these patterns from legacy 8-bit encodings to UTF-8 and put them in a single place, the package called `hyph-utf8`. We also took the opportunity to clarify the legal situation of the patterns for a few languages, as some files had not been released under clear licence terms, and we made contact with other communities who had an interest in TEX hyphenation patterns, thus creating some fruitful collaborations.

Eight years on, the project is still alive and well, and responding to internal as well as external challenges. It has long outgrown its original purpose and could be slowly turning into the central hub for hyphenation data for all free software, hence putting the spotlight on TEX on this — admittedly very specific — typographical feature. We will give a summary of the current situation, outlining the challenges we're facing, and we'll share our future plans, that will hopefully be as exciting to the audience as they are to us.

Arthur Reutenauer, Mojca Miklavec

The TEX Live M sub-project

TEX Live is the most versatile of TEX distributions, available on a variety of platforms, and very actively developed. It is the basis for MacTEX on Mac OS X, and is bundled by many package managers in Unix distributions. It has, however, a major drawback: its titanic size. This talk will discuss a sub-project to address that, with time for general discussion on wishes and ideas for TEX Live's future.

At its inception in 1996, it was contained in a CD and started growing immediately. Packages are rarely removed, due to compatibility considerations, and only technical considerations are taken into account when considering new packages: if a package fits the requirements, it is added. Today, the `texlive-full` installation scheme includes over 140,000 files and has an installed size of over 4.5 GB.

This situation is a problem for many downstream package developers and also affects the TEX community as a whole. We have started a conversation to see how we could help users find packages. We would like to offer an option to have a more controlled set of packages, probably by creating a new TEX Live "scheme" in the existing distribution by selecting among the 3200+ (to date) packages. We could define strict dependencies between packages, and also strive to do some measure of quality control, in order to create a distribution that's truly useful for newcomers and long-time users alike. The selection has to be community-driven, but there has to be a selection.

In another area, we also want to improve how the binaries are built: at the moment, they're compiled once per year by a number of volunteers who work on one or more of the twenty or so different platforms, and never get updated during the year. While this strikes a good balance between stability, the demand for reasonably recent binaries, and the workload of volunteer builders and packagers, we thought we could do better.

We have recently set up a build infrastructure that can automatically build TEX binaries after every source change for a number of platforms, send emails when builds break, show reports, and make the binaries available to users. This approach takes a lot of burden off the shoulders of people previously responsible for building TEX binaries, while at the same time giving us freedom to run the builds a lot more frequently, getting binaries to users much faster and providing earlier feedback about problems to developers. This part is almost ready and we will give some technical details of how it works.

Herbert Schulz

Workshop: TeXShop tips & tricks

An interactive workshop for users of TeXShop who want to get to know about some of the lesser-known but useful features of that front end to a TEX distribution on the Mac.

Michael Sharpe

New font offerings — Cochineal, Nimbus15 and LibertinusT1Math

Cochineal is a fork of an oldstyle font that began life as Crimson Text by Sebastian Kosch in 2010. His most recent version (2014) is named simply Crimson. LATEX support for Crimson was provided earlier this year by Bob Tennent. Cochineal tries to fill in some of the gaps in Crimson, which are most apparent in Bold and Bold Italic Greek and Cyrillic. I added a total of over 500 glyphs to improve coverage. I also added a number of tables to the OpenType versions to improve functionality in Unicode versions of TEX. Mathematical support is available via the `cochineal` option to `newtxmath`.

Nimbus15 is a reworking of the 2015 distribution of URW's Nimbus fonts. An earlier version of these fonts (from 1999) was the basis for Times, Helvetica and Courier as they appear in the TEX Live and MiKTEX distributions. What is novel in the 2015 distribution from Artifex, makers of Ghostscript, is the addition of Greek and Cyrillic alphabets to all three Nimbus fonts. The spacing and kerning of the new glyphs is quite poor in Serifed (Times) and Sans (Helvetica), and required considerable effort to correct. At the same time, I modified some of the accent shapes to more traditional ones and added glyphs to the Greek and Cyrillic alphabets so that more or less complete LGR and T2A encoded fonts would be available to LATEXers. As URW's Courier clone is not very useful in its standard form because it is much too thin and much too wide, I added a new weight between "normal" and bold, as well as a narrow version that seems to me to be neither much too thin nor too wide, and contains Greek and Cyrillic alphabets.

The recent Libertinus, Khaled Hosny's OpenType fork of Libertine with corrections and additions, contains a math font named LibertinusMath which is available only using Unicode TEXs. LibertinusT1Math reworks the math font into Type1 fonts and support files for use in LATEX. This was not as straightforward a project as it might sound, as there is no automated scheme to handle even part of such conversions. In particular, the math table in an OpenType font is a binary object without a simple text representation, and the methods used in OpenType math to make extensible math glyphs are quite different from those used in traditional TEX math fonts.

Matthew Skala

Astrological charts with `horoscop` *and* `starfont`

TEX should create beautiful documents for all fields of human endeavour, and in this talk, I describe one which is frequently under-served by typesetting systems: astrology. Astrology has its own tradition of written knowledge and symbolic notation, analogous to that of mathematics but arguably even older; and like mathematics, astrology presents unique challenges for typesetting. Writers of astrological software often focus their attention primarily on the calculations, leaving any kind of graphical presentation as an afterthought. In this talk I present the `horoscop` and `starfont` LaTeX packages, meant for creating visually appealing astrological documents using TeX. No prior knowledge of astrology, and not too much of TeX, will be assumed.

David Tulett

Development of an e-textbook using LaTeX and PStricks

For a course on decision modeling (linear, integer, and goal programming, networks, and decision trees) I created an e-textbook PDF using LaTeX and PStricks. I will discuss why I chose these programs, how I used them, and why I decided to make the final product “open access”. The entire PDF can be downloaded from `http://stor.mun.ca/handle/123456789/37463`.

The author is grateful for funding from a Teaching Fellowship to attend the 2016 TUG conference.

Boris Veytsman

Making ACM LaTeX styles

The Association for Computing Machinery is one of the largest publishers of computation-related texts in the world. It publishes more than fifty journals and even more conference proceedings every year. It was among the early adopters of TeX.

Unfortunately, over the years ACM’s TeX styles accumulated many patches and haphazard changes. They diverged to the point where reasonable support became an impossible task. This warranted a complete refactoring.

This talk discusses the experience of rewriting ACM styles and the lessons learned.

David Walden

Some notes on the history of digital typography

Over the past several years, I have been thinking about the steps in the history of digital typography, particularly with regard to desktop typesetting. In this presentation, I will share my current sketch of some of the historical steps. This may be interesting to attendees who have not tried to explicitly organize their own understanding and memory of the history of digital typography.

During the presentation period, I will also solicit from the audience steps that they think are missing from my sketch and ideas for better ways to organize the steps. I will also welcome such thoughts as I prepare my presentation; see `http://dw2.tug.org/digitype`.

Towards An Operational (La)TeX Package Supporting Optical Scaling of Dynamic Mathematical Symbols.

Abdelouahad BAYAR
Cadi Ayyad University
Ecole Supérieure de Technologie de Safi
(High college of technology of Safi)
Morocco
a.bayar@uca.ma

Abstract

In processing of digital documents containing mathematical formulas, the handling of dynamic mathematical symbols is still a big and hard problem. In fact, a tool to compose mathematics must support the typing of variable-sized symbols taking care of optical scaling and allowing to reach the known quality of metal typesetting. Until now, there is no tool that gives these possibilities in a direct and operational way. This contribution highlights and puts in practice the basic steps to develop a (La)TeX package directly based on a parametrized Type 3 PostScript font. This package will present to (La)TeX end-users a tool to compute in the usual way mathematical formulas consisting of dynamic mathematical symbols and taking into account the optical scaling. By the way, formatting (La)TeX documents, using this package, is achieved without requirements of special environments nor external programs. The concept of using parametrized Type 3 fonts directly with (La)TeX commands can give an accurate and straightforward way to manage dynamic graphics in documents formatted under (La)TeX like logo graphics for example.

Keywords: (La)TeX, PostScript Type 3, Dynamic Mathematical Symbols, Optical Scaling

1 The problem

1.1 Class of mathematical symbols

Mathematical formulas are built upon static symbols and/or variable-sized ones. Using a font in a given size, the dimension and shape of a static symbol remain unchanged in all the document. α and $+$ are good examples to represent this class. A variable-sized symbol varies in terms of size and sometimes shape from one context to an other in the same document. As example, we can cite the width hats symbols materializing angles: $\widehat{A}$ and $\widehat{AOB}$. The managing of variable-sized symbols, which we will call sometimes, dynamic mathematical symbols[1] is still a big challenge in the area of document processing (see later).

1.2 Optical scaling

This is a concept used to handle different bodies in the same font. It is opposite to linear scaling. To get the body 48 using simply the linear scaling, the encoding characteristics of characters in the body 12 are magnified four times. This is not the way used when taking care of optical scaling. The building of the character is done with tacking into account the eye of the reader. More details on optical scaling concept are found in [3, 6, 7]. When we consider Humans or trees fro example, we can see obviously that they do not grow in a linear model. The human eye is specifically satisfied of in an art view point. It would be better to talk about *natural scaling* than optical scaling since the first is more general than the second. By the way, the confusion between "optical scaling" and "optical scale" introduced by Harry Carter in typefounding [1] will not happen.

1.3 Metal/Digital Typesetting and optical scaling

In old books, especially those in mathematics, mathematical formulas were typeset with respect to optical scaling. We give an example of a formula taken from [8] exhibiting the metal braces. It is clear that these braces are not related with a linear scaling (See Figure 1). Optical scaling was not difficult to reach since symbols are processed in their final sizes. However, the support of optical scaling is not easy to perform in automatic systems computing symbols or fonts especially in the case of digital typesetting. More information on this point is in [3, 5]

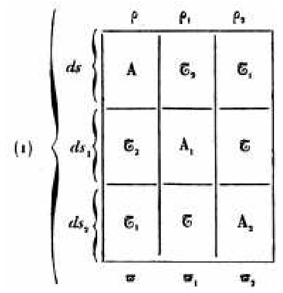


Figure 1: Braces in metal typesetting

[1] "Dynamic Mathematical Symbol" term encompasses variable-sized symbols and also symbols defined in dynamic fonts. Dynamic fonts are in reality fonts in which the character graphics are defined at each instantiation in the printing time and not in the definition of the font.

1.4 Existing works

Dynamic characters, especially dynamic mathematical symbols, are omnipresent in scientific documents. So, a good application to process scientific documents must be made up of all required means to support dynamism characteristics. Along these four decades, developed tools have supported dynamism in different ways. We can cite in this case native TEX [9] and LATEX [10] to support mathematical variable-sized symbols. One important tool to indicate is Curext [11] which is a (La)TEX package. With this package, it is possible to typeset mathematical formulas consisting of variable-sized mathematical symbols particularly Arabic ones. In the same way, Curext allows to write Arabic text taking into account the Kachida concept. We have to note that Kachida materializes a important phenomenon of dynamism in Arabic text typesetting. Detailed information about Kachida and justification of Arabic texts can be found in [13]. It is very important to consider the work accomplished in [2, 3]. It consisted of the design of math-fly font, a PostScript font Type 3 to supply dynamic mathematical symbols tacking care of optical scaling. The particular property of this work, in comparison with the preceding, is that it is not used under nor with (La)TEX.

Curext as a package to extend the TEX capabilities in handling variable-sized symbols has some difficulties in processing mathematical formulas containing more than two (matched) dynamic symbols [11]. So, it does not offer the adequate support to manage general cases of mathematical formulas.

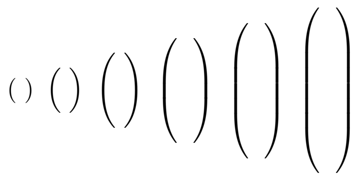


Figure 2: Stretches of parentheses in TEX

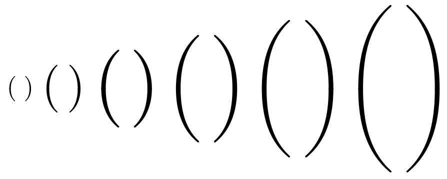


Figure 3: Stretches of parentheses via PostScript

As regards TEX, everybody knows that it supports the composition of mathematical formulas with multiple variable-sized symbols. The optical scaling is not very well supplied since the thickness

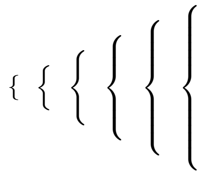


Figure 4: Stretches of left brace in TEX

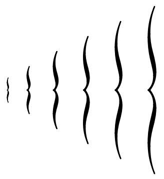


Figure 5: Stretches of left brace via PostScript

cannot be changed after a certain level (See Figure 2). Furthermore, due to non dynamic properties of metafont, dynamic variable-sized symbols in big sizes change in shape (See Figure 2) and do not look like the metal ones as showed in Figure 3. For some variable-sized mathematical symbols as braces for example, TEX does not allow to typeset them in the format of metal symbols neither in small sizes nor in big ones. Figure 4 and 5 highlight the state.

About optical scaling, we notice that any one of these tools is completely operational. *The problem of typesetting mathematical formulas with good quality respecting optical scaling is still challenging.*

In the following, the paper follows the plan: the second section presents what is required for handling dynamic mathematical symbols with taking into account the optical scaling. In the third section, the practical and operational way to design the system is given. The next section is dedicated to describe and compare the implementation of the package under different TEX tools. The paper ends with conclusions and perspectives.

2 The Requirements to handle dynamic mathematical symbols taking care of optical scaling

To realize a convenient tool to compose mathematical formulas based on dynamic mathematical symbols with respect to optical scaling, it is required to subdivide the study into two parts. The first one concerns the *font of symbols* whereas the second refers to the *way to use this font* to produce mathematical formulas. In all cases, we must not neglect the fact that the tool has to assist the (end-)user to produce good mathematical formulas in direct and straightforward way.

We know that fonts are classified in two groups: static fonts and dynamic fonts. We give briefly and accurately the difference between the two classes. In static fonts, shapes (the graphic to print) of characters are generated and finalized before the printing time. However, in dynamic fonts, characters take their printing characteristics at printing time. More details and examples for comparison are found in [4]. A suitable font to deal with variable-sized symbols must be dynamic but in more it has to get the following features:

- The language to implement programs encoding dynamic symbols must provide more flexibility in parametrizing symbols. Also, it must have the ability to receive values from out of the font to instantiate parameters and so generate the shape to print.
- The interaction between the document processing system and the font language must be well defined and directly used in the document processing task.

(La)TEX, as a text formatting system, provides natively an interface to fonts encoded in metafont language. This is achieved principally via tfm files. Nevertheless, metafont does not allow manipulating dynamism at printing time. (La)TEX uses other kinds of fonts like Postscript Type 1, True Type font or the hybrid of the two previous types OpenType. These fonts are referenced by TEX as if they were virtually metafont fonts. So the use of these fonts does not add either a mean to supply real dynamism. It is also very interesting to cite XeTEX and XeLATEX. These are extensions to TEX and LATEX in order to work directly with OpenType fonts (also Type 1 and TrueType) without use of any intermediate mapping files. Even with this capacity, XeTEX and XeLATEX do not support a complete dynamism due to the limited interaction interface between The TEX engine and the font. Furthermore, OpenType support only a semi-dynamism or a discrete dynamism.

PostScript Type 3 are fonts with some particularities:

- They use full PostScript language. This means that the specification of fonts can use all operators and constructors existing in the PostScript language especially local and global variables. Variables are the mean to communicate new characteristics to the procedure encoding dynamic symbols.
- The concept of caching the character bitmaps (used in Type 1) can be deactivated via replacing `setcachedevice` with `setcharwidth`. This implies that each time a given character is to be printed, its bitmap will be fully computed. Consequently, the model supporting variability (Optical Scaling) can take new values and states.

The PostScript font Type 3, further to that, loses some important abilities like fast printing, improvement via hints and handling by Adobe Type Manager (ATM). But the support of dynamic mathematical symbols taking care of optical scaling outweighs the disadvantages. Also, nowadays, the computers inside printers are so fast and so large that the time to move the paper inside printers dominates the printing speed. Moreover, the industry of printers meets day after day important improvements in resolution. Then, The efficiency benefits such as caching and hints are gone.

PostScript font Type 3 can be used by TEX in the same way as used Type 1. In this case, we cannot derive the benefit of dynamic specification in PostScript. In [2], the authors stated that a parametrized font Type 3 could not be fully used by formators (editors) such as TEX or other without modifications of the way they call formula symbols. Due to this, they chose to check their font in the Grif project. In our case, the font PostScript Type 3 will be inserted directly in the (La)TEX source of the document to format thanks to `\special` macro. Of course, the way to process dynamic mathematical symbols in mathematical formulas will be reviewed. The details of the concept is given in the sequel.

3 The design of a practical and operational system

3.1 General package layout

The development of a package able to use directly a PostScript font Type 3 is based on the existing possibilities of interaction between (La)TEX and PostScript programs. This is done using the command `\special` via the dvips driver to translate dvi files to Postscript ones [15]. Precisely, we use the ways to include literal PostScript in TEX documents to work with the font PostScript Type 3. A summarized design of the package is given bellow.

- The PostScript font Type 3 supporting dynamic mathematical symbols and all useful procedures are defined in the package as a literal header: '!' `\special`. This is mandatory since the font will be used later when including other PostScript codes to show Postscript dynamic symbols. Our font Type 3 is named "`dynMath`". The command is : `\special{! ... specification` `dynMath...}`. The font implements the mathematical symbols to support curvilinear stretching depending on the values of two global variables[2] h and w. In the font, the stretching model allows symbols to stretch in height (depth) and width depending on the values of h and w and keeping the same thickness. We can remark that the scaling is not linear. It is a semi-optical scaling since the thickness is not affected. It is done in the defined macros like `\meLeft` for example (see later).

- The principal macro in handling mathematical formulas and so dealing with inclusion of the PostScript dynamic mathematical symbols is defined in the package. Its skeleton is: `\def\meLeft#1#2\meRight#3{...}`.
  `#1`: the left delimiter,
  `#2`: the formula to delimit and
  `#3`: the right delimiter.
  This macro manages dynamic mathematical symbols which are delimiters. Regarding the other dynamic symbols like 'radical' for example, they are defined in separate macros or in some cases the existing macros will be redefined. About the delimiters, we chose to finalize at the moment the implementation only of two symbols namely parentheses and braces. In reality, theses two groups of symbols are an *adequate representation of curved symbols*. Parentheses have simple curved shapes whereas the braces have curved shapes with inflections. We have to notice that variable-sized symbols that are simple combination of lines are very simple to implement in the font.
- In `\meLeft` macro:
  1. The dimensions; width, height and depth of the formula are computed. Let h_f, w_f and d_f be these dimensions respectively.
  2. Depending on h_f, w_f, d_f and the left delimiter symbol, `\meLeft` determine the stretching amounts of h and w. Then, the corresponding body `fs` in which the font "`dynMath`" will be used to write the left symbol is calculated.
  3. The dimensions of the left symbol `symHeight`, `symWidth`, `symDepth` tacking into account the PostScript font `fs` are determined.
  4. In an horizontal box using `\hbox` of dimensions `symHeight`, `symWidth`, `symDepth`, the left symbol is written using a literal PostScript.

```
... \special{" ...
/fs ... store
/h ... store
/w ... store
/dynMath findfont fs scalefont
setfont
<code of The symbol> show
}
...
```

  5. The formula is written.
  6. The steps from the second to the fourth are applied for the right delimiter. Frequently, `symHeight`, `symWidth` and `symDepth` remain unchanged.

[2] We have chosen to give simple names h and w to make easy the processing of the equations in the paper. In final work, meaningful names will be used. for example h and w will be replaced by `verticalStretch` and `horizontalStretch` respectively.

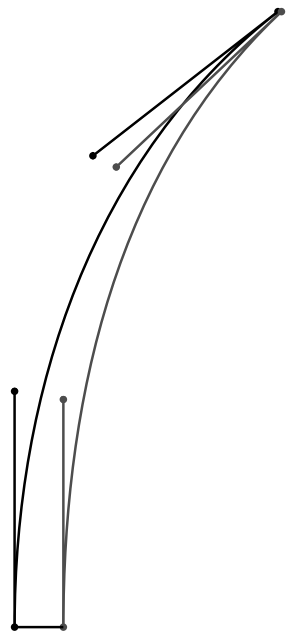


Figure 6: Top Part of left parenthesis - Initial encoding in body 500

3.2 The design of `dynMath` font

The font `dynMath` is an image of the font "cmex10.mf". The simple difference is that a symbol appears in `dynMath` only once oppositely to the case of "cmex10.mf". For example, the left parenthesis is encoded in cells numbered 0, 16, 18, 32. The stretchable parenthesis is built upon the characters numbered 48, 66 and 64. However, in `dynMath`, only a parametrized parenthesis is located in the font at the order 0. For some particular values of the parameters, we get the parenthesis with expected characteristics. As said before, only (,), ⎫and ⎧with code numbers 0, 1, 8 and 9 respectively are encoded for the moment. We used the existing font "cmr10.mf" to get embryos of left and right parentheses which we parametrized applying a mathematical model. Notice that we did not use "cmex10.mf". This is because the small parenthesis in "cmex10.mf" is bigger than the normal one (parenthesis in text). About left and right braces, we know that any one of the metafont fonts given with TEX distributions supports the braces looking like the metal ones. So, we integrally designed them. The command applied to generate the basic encoding of parenthesis through "cmr10.mf" is:

```
mpost '&mfplain \mode=localfont; \
mag=100.375; input cmr10.mf'
```

To explain the global concepts to design the font, we use the parenthesis as instance. The same process is applied to the other symbols. Without loose of generality, we consider the showing only for the top part of the left parenthesis (with respect to the mathematical axis).

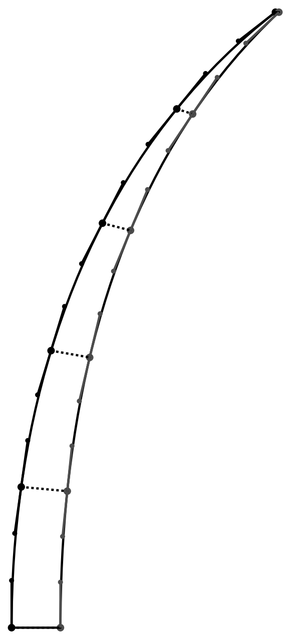


Figure 7: Decomposed Top Part of left parenthesis (Parametrized curves) in body 500 without stretching

Applying metapost à "cmr.mf", we get the encoding of left parenthesis. Its top left part is showed in Figure 6. It is defined based on two Bézier curves linked by two line segments in top and bottom. To get a parenthesis symbol being able to stretch when needed, the two Bézier curves are multi-decomposed using the generalized algorithm of refinement [16].

For clear explanation, we consider a refinement of the fifth order. In Figure 7, the curves are decomposed according to the decomposition parameters $1/5$, $1/4$, $1/3$ and $1/2$. Every Bézier curve of the encoding appears as a concatenation of five subcurves. The latter, are then parametrized tacking into account the variables h and w such that when the stretching amounts are equal to zero the shape is identical to the initial one illustrated in Figure 6.

Figure 8 shows an example of stretching in body 500. h and w take the values 250 and 83.33 PostScript points respectively ($83.33 \approx 250/3$). We notice that initial and stretched versions of the half symbol differ in height and width but they have the same thickness. To emphasis this fact, let us consider the line segments joining the extreme control points of the left and right sub-curves. Each segment in Figure 7 and its correspondent in Figure 8 are parallel and have the same length. It is obvious that the scaling is not linear. It does not support a right optical scaling either since the thickness remains unchanged. In the PostScript font, only a semi optical scaling is defined. The optical scaling is completed in TEX package. The way to parametrize the control points in order to support this semi optical scaling obeys to a strict mathematical model. We would not present mathematical concepts here because this is outside the objective of this paper. Nevertheless, we cite the basics of the development. The curves are parametrized based on the mathematical model which insures that stretched and initial curves have the same geometric and similarity characteristics. In our PostScript font, the left part of the parentheses has undergone a Bézier refinement of order 15. Whereas in the example, the fifth order has been enough because the amounts of stretching are not big.

3.3 Optical scaling support

In this section, we present how the optical scaling is supported by the package. The best way to describe the concepts is via the delimiters, particularly balanced ones as parentheses and braces. This can not be done without enumerating the principal characteristics of a mathematical formula.

We consider an abstract mathematical formula to present the characteristics. It consists of a box with some height, depth and width (which is not interesting in this paper). Figure 9 and Figure 10 show the two cases of mathematical formulas, id est

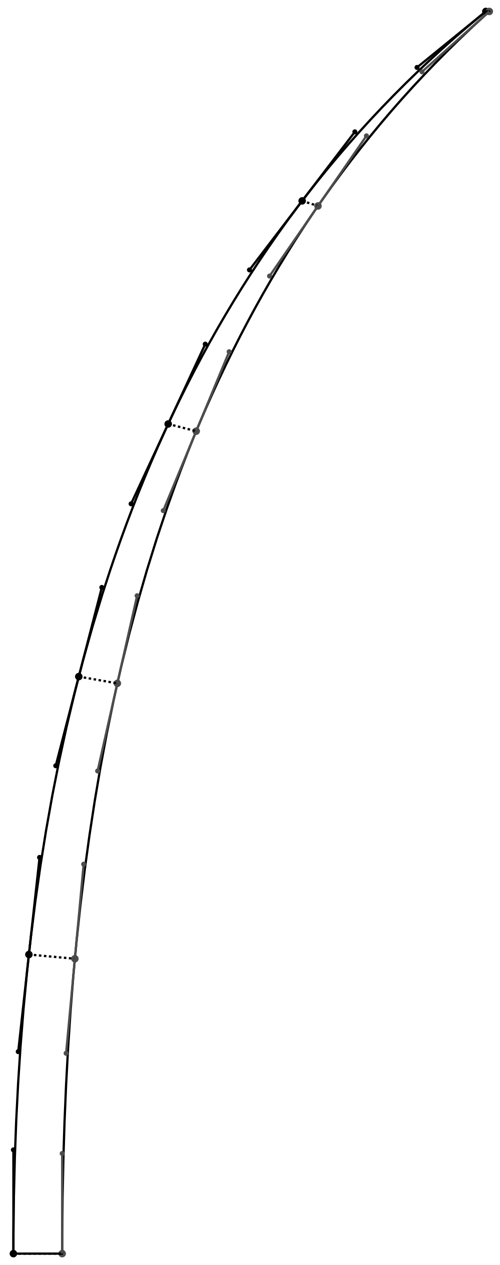


Figure 8: Decomposed Top Part of left parenthesis (Parametrized curves) in body 500 stretched 250 vertically and 83.33 horizontally

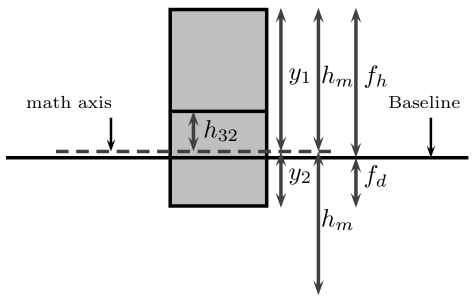


Figure 9: Abstract high mathematical formula

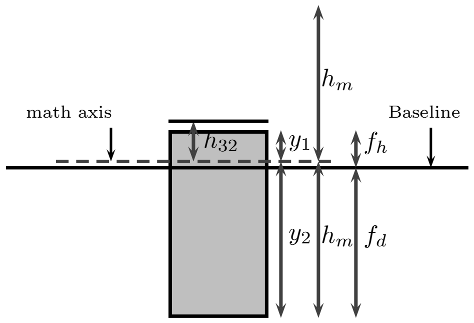


Figure 10: Abstract deep mathematical formula

when the formula is high or deep. Moreover, they introduce some characteristic variables of formulas:

- f_h: height of formula from the baseline.
- f_d: depth of formula from the baseline.
- y_1: mathematical height of the formula. It is measured from the mathematical axis to the top of the formula.
- y_2: mathematical depth of the formula. It is measured from the mathematical axis to the bottom of the formula.
- h_m: mathematical balanced height (depth) of the (balanced) formula. We have that $h_m = \max(y_1, y_2)$.
- h_{32}: the mathematical height of parenthesis in body 32 (corresponding to the height h^p_{32} in PostScript DynMath - of course the value manipulated in the package considers the relation between pt and bp). h_{32} is a reference in processing the optical scaling (see later).

We note that a TEX variable v_n is the value in TEX unit corresponding to v^p_n in PostScript unit. For example, h_{32} is the hight in pt corresponding to h^p_{32} being the hight in PostScript of the dynamic symbol in body 32. We have formally $v_n = 1.00375 \times v^p_n$.

A part of optical scaling, as previously said, is supported by the PostScript Type 3 font whereas the other part is directly a job of the TEX package.

After a study of the fonts generated from "cmr.mf" and "cmex10.mf" via metapost, we noticed that the standalone parentheses symbols (Symbol compound of one character) got from "cmex10.mf" are in some way linear scaling of the standalone parenthesis supplied by "cmr10.mf". The dimensions of the big parenthesis in "cmex10.mf" is approximately three times equal to the ones relative to the parenthesis in "cmr10.mf". Consequently, the optical scaling is handled into two different ways relating to the value of h_m. The first case deals with the values of h_m less or equal than h_{32} whereas the second takes care of values greater strictly than h_{32}.

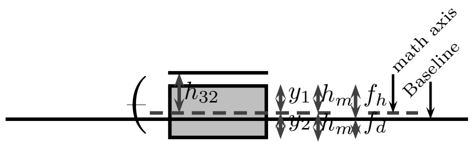


Figure 11: Abstract mathematical formula with $h_m \leq h_{32}$ and non aligned math axis

When the mathematical height h_m is less or equal than h_{32}, the optical scaling is treated simply as a linear scaling. Figure 11 illustrates this case. Let fs be the PostScript font size in which the height of the half of the left parenthesis (taken as example) equals h_m. Then the font `dynMath` is set to fs and the delimiter is written in a `\special` macro.

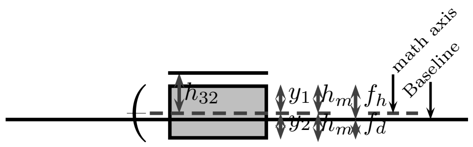


Figure 12: Abstract mathematical formula with $h_m \leq h_{32}$ and aligned math axis

We can see that when the delimiter is introduced, the mathematical axis of the formula and the delimiter's one are not aligned. This is normal because the "TeX body" in which the formula (10, 11, ...) is written will be usually different to the delimiter body. in `\sepecial` macro, a shifting operation is accomplished before writing the PostScript delimiter (See Figure 12)

In the case where h_m is strictly greater than h_{32}, the PostScript font size is managed differently. The mathematical model adopted to formalize stretching of mathematical symbols supplied us a maximal vertical stretching of 32700 bp. As The body 32 is a threshold to handle the symbol stretching, we consider the maximal amount allowed in stretching the half of parenthesis to be h^p_{max}. The superscript p relates to Postscript context. We have:

$$h^p_{max} = \frac{32700 \times 32}{1000} = 1190.4 \tag{1}$$

Let h_{max} be the equivalent amount in TeX points then we have

$$h_{max} = 1.00375 \times h^p_{max}\text{pt} = 1194.864\text{pt} \tag{2}$$

In the sequel, we would give needed dimensions only in TeX points since all calculations are done by TeX. The first step in processing is to determine the body that will be used to compute the delimiter. The PostScript font `dynMath` embeds the possibilities of extensions of symbols but without affecting the thickness. This is what remains to support the optical scaling. Let e be the thickness variable (in TeX units). e is calculated as a function of the mathematical height of the formula to delimit. This is given in Equation 3.

$$e\left(h_m\right) = c_1 h_m + c_0 \tag{3}$$

Where c_1 and c_0 are constants satisfying the following requirements:

- $e\left(h_{32}\right) = e_{32}$
- $e\left(h_{max}\right) = \lambda \times e_{32}$
- e_{32} : thickness of the dynamic symbol in body 32.
- λ : a scaling constant factor[3]. We notice that λ is not a global value for the font. It depends on the dynamical symbol.

for λ, e_{1000} and e_{32} known, we can determine the body fs for which the thickness of the symbol is e. Using fs, we can produce the corresponding math height of the symbol h_{fs}. The formulas giving these variables are in Equation 4 and 5.

$$fs = \frac{1003.75}{e_{1000}} e \tag{4}$$

$$h_{fs} = \frac{h_{1000}}{1003.75} fs \tag{5}$$

[3] For parenthesis and brace, the satisfaction is reached with $\lambda = 3.236pt$. 3.236 equals 2×1.618. 1.618 is the golden number

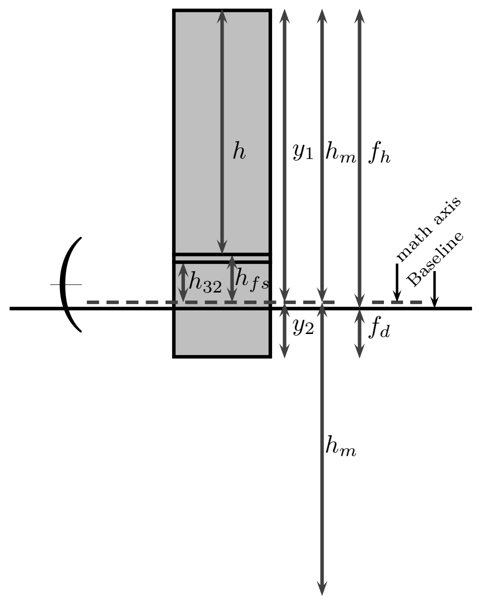


Figure 13: Abstract mathematical formula with $h_m > h_{32}$, non aligned math axis and non stretched parenthesis

As an important remark, we can see easily that $h_{32} < h_{fs} < h_m$. Once h_{fs} is processed, the amount of vertical stretching h can be determined let is $h = h_m - h_{fs}$ (See Figure 13).

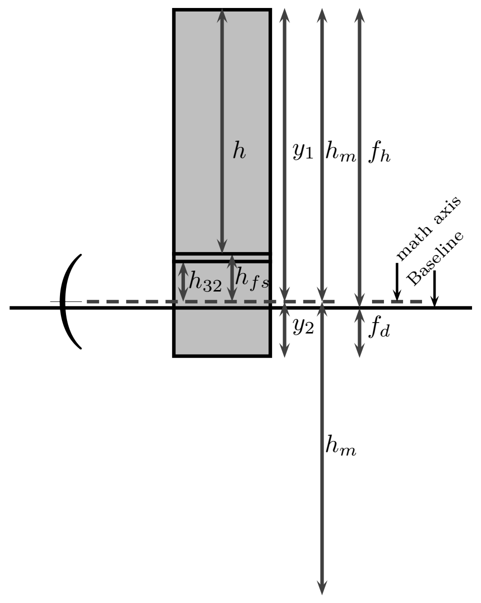


Figure 14: Abstract mathematical formula with $h_m > h_{32}$, aligned math axis and non stretched parenthesis

The dynamic symbol is written, especially the left parenthesis as showed in Figure 13, in `\special` macros using the font fs^p ($fs^p = 0.9962 \times fs$) as body. We remark, like for the case where $h \leq h_{32}$, that the math axis of the formula and the parenthesis's one do not coincide (See Figure 13). So, a shift transformation is applied before writing the symbol in PostScript (See Figure 14).

The horizontal stretching amount w may take values depending on h. In the case of the left parenthesis, w describes the interval $[0, {}^{h}\!/_{3}]$. In Figure 15, the parenthesis in fs body is stretched of h vertically and $w = {}^{h}\!/_{3}$ horizontally.

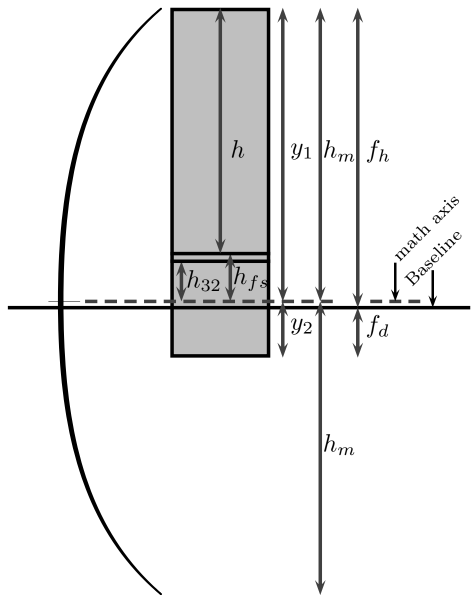


Figure 15: Abstract mathematical formula with $h_m > h_{32}$, aligned math axis and stretched parenthesis

4 Implementation

As we said, the system to typeset mathematical formula based on dynamic variable-sized symbols is composed into a PostScript Type 3 font and a set of TeX macros. Until now, the package is presented as a simple TeX source file. It contains only TeX macros that are recognized also under LaTeX. So the package operates with TeX and LaTeX. In the same time, it is a good embryo to develop a LaTeX package. One of the important steps in the execution process of the macros managing variable-sized symbols is the catching of the current mathematical style. It is a mandatory task in order to get the right dimensions of the mathematical formula and determine the true sizes of the dynamic mathematical symbol. First, we have developed a package that can operate in allTEX programs. The determination of the current math style imposes the use of complete recursion, id est recursion in macros definitions and callings. So the package is very slow and more hungry in memory. To solve this problem, we resorted to the use of luaTEX and luaLATEX since they supply `\mathstyle` (`\luatexmathstyle`) which permits the catching of the mathematical style on the fly. Then, the time of processing and the use of memory are very reduced. Of course, since a PostScript Type 3 font is used conjointly with the TEX package, then documents composition sources are formatted via `dviluatex` (`dvilualatex`) and `dvips` commands.

5 Conclusions

We developed a mini-package TEX offering the support of mathematical variable-sized symbols with respect to the optical scaling. As illustration, we implemented only two symbols which are the parenthesis and the brace. But the support of these two symbols proves well the feasibility as well as the possibility to produce scientific documents in quality of metal typesetting. As perspective, we will finish, in the font and package levels, the support of all the rest of dynamic mathematical symbols. In the same time, some options enabling effects on the printing quality will be added to the final package. A more important task will concern the optical scaling but with considering an artistic view point. Indeed, the relationship between the values of horizontal, vertical stretching and the thickness will be studied tacking into account the artistic satisfaction.

References

[1] Carter, Harry. ”The Optical Scale in Typefounding”, Typography, No, pp.2-6, Autumn, 1937.

[2] Jacques André, Irène Vatton, *Contextual Typesetting of Mathematical Symbols Taking Care of Optical Scaling*, Technical report No 1972, INRIA, October 1993.

[3] Jacques André, Irène Vatton, “Dynamic Optical Scaling and Variable-sized Characters”, *Electronic Publishing,* Vol 7, No 4, pp. 231-250, December 1994.

[4] Jacques André, B. Borghi, “Dynamic Fonts”, *PostScript Language Journal*, Vol 2, no 3, pp. 4-6, 1990.

[5] Richard Southall, “Character description techniques in type manufacture”, in Raster Imaging and Digital Typography II, eds., Robert A. Morris and Jacques André, pp. 16–27, Cambridge, UK, (October 1991).

[6] Circuitous Root, “From the Optical Scale to Optical Scaling”, http://www.circuitous-root.com/artifice/letters/press/typemaking/mats/optical/index.html, 2016.

[7] Circuitous Root, “Clubs and Cults Revisiting the Concept of "Typeface" and the Optical Scale in Typefounding”, http://www.circuitousroot.com/artifice/letters/press/typemaking/making-matrices/terms/logical-grouping/clubs-and-cults/index.html, 2016.

[8] G. Lamé, “Leçons Sur Les Coordonnées Curvilignes Et Leurs Diverses Applications”, Imprimerie de Mallet Bachelier, Paris - Rue du Jardinet 12, 1859.

[9] D.E. Knuth, *The TEXBook, Computers and Typesetting*, Reading MA : Addison-Wesley, Vol.A, 1984.

[10] Leslie Lamport, *LATEX-A Document Preparation System*, Reading MA: Addison Wesley, 1985.

[11] Azzeddine Lazrek, “CurExt, Typesetting variable-sized curved symbols”, *EuroTEX 2003 Pre-prints : 14th. European TEX conference,* Brest, France, pp. 47-71, 2003.

[12] M. J.E. Benatia, M. Elyaakoubi, A. Lazrek , “Arabic Text Justification”, *TUG 2006 Conference Proceedings*, Volume 27, No 2, pp. 137-146, 2006.

[13] Mohamed Elyaakoubi, Azzeddine Lazrek, “Justify just or just justify”, Journal of Electronic Publishing, Volume 13, Number 1, 2010, ISSN 1080-2711 (http://dx.doi.org/10.3998/3336451.0013.105)

[14] Daniel M. Berry, “Stretching Letter and Slanted-Baseline Formatting for Arabic, Hebrew and Persian with dittroff/ffortid and Dynamic PostScript Fonts”, *Software-Practice and Experience*, vol. 29, no. 15, pp.1417-1457, 1999.

[15] Tomas Rokiki, “Dvips: A DVI-to-PostScript translator”, Dvips User Manual - version 5.996, https://tug.org/texinfohtml/dvips.html, 2016.

[16] Brian A. Barsky, *Arbitrary Subdivision of Bézier Curves*, Technical Report UCB.CSD 85/265, Computer Science Division, University of California, 1985.

MFCONFIG : METAFONT plug-in module for Freetype rasterizer

Jaeyoung Choi, Sungmin Kim, Hojin Lee and Geunho Jeong

Abstract

One of the advantages of METAFONT is its ability to show a variety of font styles by changing of the values of the parameters that represent the font characteristics. This advantage can be applied to not only simple Roman-alphabet characters, but also to complicated CJK (Chinese-Japanese-Korean) characters. Second, the font families like bold, italic, and bold-italic do not need to be created for METAFONT, because it can automatically generate a variety of styled fonts through changing the parameter values. Therefore, METAFONT can reduce the development time and cost for the production of a font family. It is not possible, however, to directly use METAFONT in general font engines, as it must be changed to the outline-font format if it is to be used in the current PC environment. In this paper, the *MFCONFIG* module that enables a direct usage of METAFONT on Linux is proposed; although, it must be installed with the popular rasterizer Freetype. *MFCONFIG* is a plug-in module for the *FONTCONFIG* library engine, which makes the Freetype engine compatible with the current digital font types of bitmap and outline; furthermore, with the use of different parameters, the proposed module supports a variety of fonts, generated from METAFONT with different parameters.

1 Introduction

Text is an effective way to communicate and record information. With the growing use of smart devices, digital fonts are more commonly used than analog fonts. Although many styles of digital fonts have been created, they still do not meet the requirements of all users, and users cannot change digital-font styles freely[1]; for instance, if a user wants to use a thinner outline font, either he/she has to find a thinner styled font, or an in-application function to change the font thickness. As several different features of font style are needed, though, a simple searching or the changing of the font style of an existing font is not typically easy. A perfect application for the satisfaction of users' diversified requirements regarding font styles does not exist. Also, it is impossible to provide all of the styled fonts in accordance with users' preferences.

Currently, popular digital fonts of bitmap or outline have a limit to change font style[2]. However, METAFONT is a structured font that allows users to change the font style freely. METAFONT, a TeX font system, had been introduced by D. E. Knuth[3]. It has functions for drawing characters and parameters to determine the font styles. When the user changes the parameters, the font style is changed automatically. Therefore, a variety of styled fonts can be generated from one METAFONT font. Figure 1 shows a variety of styled fonts are created by the changing of the thickness, slant, and a combination of two thickness and slant styles for the alphabet “A” and the Chinese character “漢” is shown. If other features such as serif and pen are applied together, a variety of styled fonts can be more generated.

Most users, however, are unable to use METAFONT on their PCs because the current font engines do not support METAFONT. METAFONT is expressed as program code, so it is different from the general digital-font types of bitmap and outline. If a user wants to use a specific METAFONT in general font engines such as Freetype, then he/she needs to convert the METAFONT font into the corresponding outline font format.

In the case of Roman characters, the design of only several hundreds of characters is required, and their shapes are simpler than those of CJK (Chinese-Japanese-Korean) characters. In the mid-1980s, when METAFONT was introduced, the PC was not fast enough to enact a real-time conversion of the METAFONT fonts into the corresponding bitmap or outline fonts. Moreover, outline fonts have been

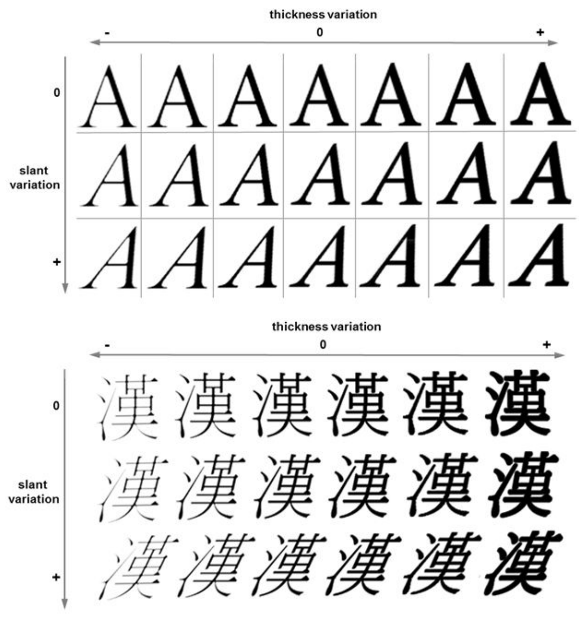


Figure 1: METAFONT style variation

more commonly used, rather than METAFONT, up until the present time.

The current PC, however, ensures good performance regarding the real-time execution of METAFONT. If METAFONT could be used directly in a PC, then users could easily make and use a variety of styled fonts by themselves. As previously we viewed, one METAFONT font can basically be represented by a variety of styled fonts by changing of the style parameter values. Therefore, METAFONT can save great amounts of time and repeated effort in terms of font design to make font families of plain, italic, bold, and italic-bold fonts. In particular, in the case of CJK-character usage, METAFONT could represent an effective way to make and display a variety of font style. Because, compared to the alphabet, CJK characters are very complicated in shape and they are expressed by combinations of radicals.

In this paper, a METAFONT module that enables a direct METAFONT usage on Linux is proposed. It is possible to plug this module into *FONTCONFIG* for providing digital font information to the FreeType engine. If the *MFCONFIG* module is used, the conversion of a METAFONT into its corresponding outline font is unnecessary. It is very simple to change font styles by applying new parameter values. Also, this module can interact with most of the existing *FONTCONFIG* functions without the modifying itself or the Freetype rasterizer. The *MFCONFIG* module therefore has good usability and compatibility regarding the support of METAFONT in the Freetype engine.

2 Research for font system

FONTCONFIG[4] provides an extended font configuration to the Freetype rasterizer, and the Xft(X-FreeType interface library)[5] had been developed to provide interfaces between applications and Freetype. These font libraries are able to collect fonts' information on the current PC system such as fonts' path, fonts' style information, and extra meta information, and so on. Figure 2 shows a font-output sequence that is required from the applications on the X Window system in Linux. After an application sends a font request according to name and style to the Xft library, it also delivers the request information to *FONTCONFIG*. *FONTCONFIG* uses its internal commands to check the following conditions: (1) Whether the requested font is installed, (2) whether the style of the user's request has been applied to the stored font, and (3) whether the requested font has already been stored with the printing format in the cache. (4) If the requested font is not stored in the cache, it needs to be converted into the requested printing format and stored in the cache, and (5) the requested font in the cache is selected and then delivered to Freetype. Lastly, the requested font is printed with the font styles.

FONTCONFIG is a library for Freetype, and it is capable of supporting general digital-font formats that can also be processed in Freetype. The architecture of *FONTCONFIG* is shown in Figure 2. It can support TrueType, OpenType, Type1, CFF, PFR, and DBF, but it does not yet support METAFONT. For the direct support of METAFONT in *FONTCONFIG*, it might be necessary to change the internal codes of *FONTCONFIG*. For instance, the changing of the overall processes in *FONTCONFIG* from fc-scan" (for font searching) to fc-pattern (for the matching of the styled font pattern). It is not a simple work, however. The Xft library interface exists between an application and *FONTCONFIG*, and it is for the providing font information such as font name and font size. It is not a good approach to modify Xft support METAFONT. It might reduce Xft's performance.

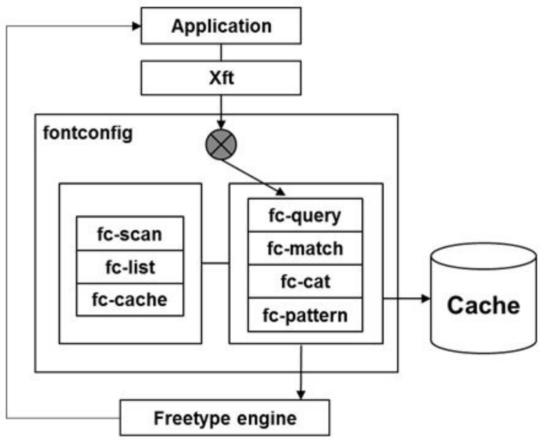


Figure 2: Architectures of fontconfig

VFlib[6][7] is a font driver system for the supporting variety of font types. The system supports virtual fonts like BDF, PCF, and TrueType, as shown in Figure 3. It provides database including general fonts' type information, and it also provides a useful API for the supporting variety of font types. VFlib includes separate modules for each font type, so for the support of METAFONT, a new module can be added to it. But VFlib is heavy because it consists of many different kinds of font drivers and an information dataset of default font information. In addition, the VFlib interface is required if an application wants to use the VFlib library. Therefore, if a METAFONT module is added to VFlib, additional functions must be implemented for every relevant

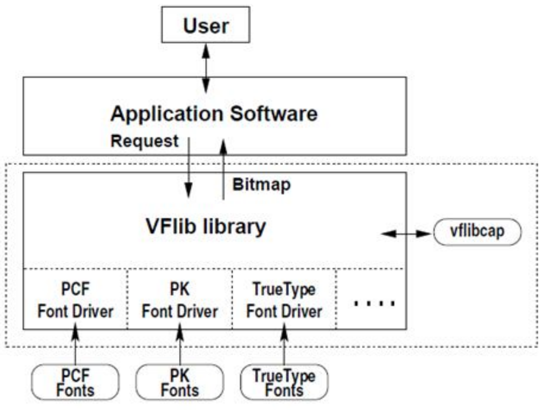


Figure 3: Architectures of VFlib

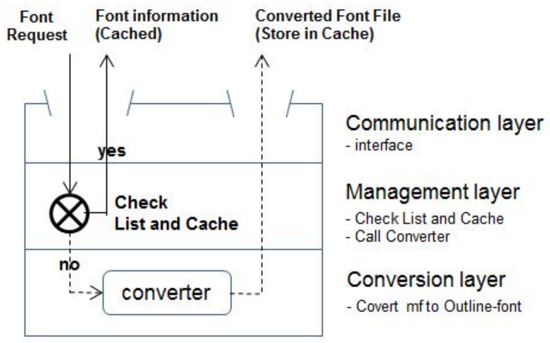


Figure 4: Three layers of *MFCONFIG* module

application. So, VFlib is not suitable to support for METAFONT.

The proposed *MFCONFIG* module in this paper combines the following two features: (1) The process for the printing of digital fonts in *FONTCONFIG*, and (2) the font driver architecture of VFlib. The module can process METAFONT independently, and it can be easily installed or removed since it is implemented as a plug-in module. Also, the steps that are used for its implementation are similar to *FONTCONFIG*'s inner commands, so METAFONT can be used with the existing digital font formats.

3 Implementation of *MFCONFIG* module

As shown in Figure 4, the *MFCONFIG* module consists of the following three layers: Communication Layer, Management Layer, and Conversion Layer. The Communication Layer provides an interface between *FONTCONFIG* and *MFCONFIG*. The Management Layer checks whether the requested METAFONT is ready in the cache. If not, it sends a request message to the Conversion Layer to convert the METAFONT. The Conversion Layer makes a new outline font file by using the requested METAFONT and the customized style values. The resulting outline font is stored in the cache.

As shown in Figure 5, *MFCONFIG* can be plugged into *FONTCONFIG*. First of all, an application requests a font from the Xft library (step 1). Next, *FONTCONFIG* sends the font information and the values of the style parameters to *MFCONFIG* through the interface of the Communication Layer (step 2). This interface checks if the requested font is a METAFONT font or not.

In the case of METAFONT, *mf-query* analyzes the requested information, and *mf-match* tries to find this METAFONT from *mf-list*. If the information does not exist in *mf-list*, the requested METAFONT font is not installed. In this case, *mf-query* returns a not found flag to *FONTCONFIG* (step 3). Otherwise, if METAFONT is installed and is already stored in the cache, *mf-query* returns a found flag to *FONTCONFIG* (step 3).

Sometimes, the requested METAFONT font is installed, but the corresponding outline font may not be stored in the cache memory yet. In this case, *mf-converter* in the Conversion Layer needs to convert the METAFONT font into the corresponding outline font (step 2-a). In this step, the METAFONT font and the styled parameter values from the application are required for the conversion. After the conversion, the outline font is stored in the cache (step 2-b), and *mf-query* sends the found flag to *FONTCONFIG* (step 3).

After step 3, the remaining steps are the default steps of *FONTCONFIG*. The font information is sent to the internal programs of *FONTCONFIG* (step 4) that try to find the corresponding outline font in the cache (step 5); then, this outline font is sent to the Freetype rasterizer (step 6). If *MFCONFIG* returns the not found flag, then *FONTCONFIG* uses a default font file. Lastly, the Freetype engine prints the outline font that is made from the requested METAFONT with the styled parameter values.

The details of the three layers in *MFCONFIG* are presented below. The Communication Layer that is an interface between *FONTCONFIG* and *MFCONFIG* is the starting point of the *MFCONFIG* module. Therefore, Freetype engine receives the font information from *FONTCONFIG* as like previously. The main functions of the interface are as follows: (1) The delivery of the requested METAFONT information to the Management Layer, (2) the returning of the results to *FONTCONFIG*, and (3) the storage of the outline-font file from *mf-converter* in the cache memory.

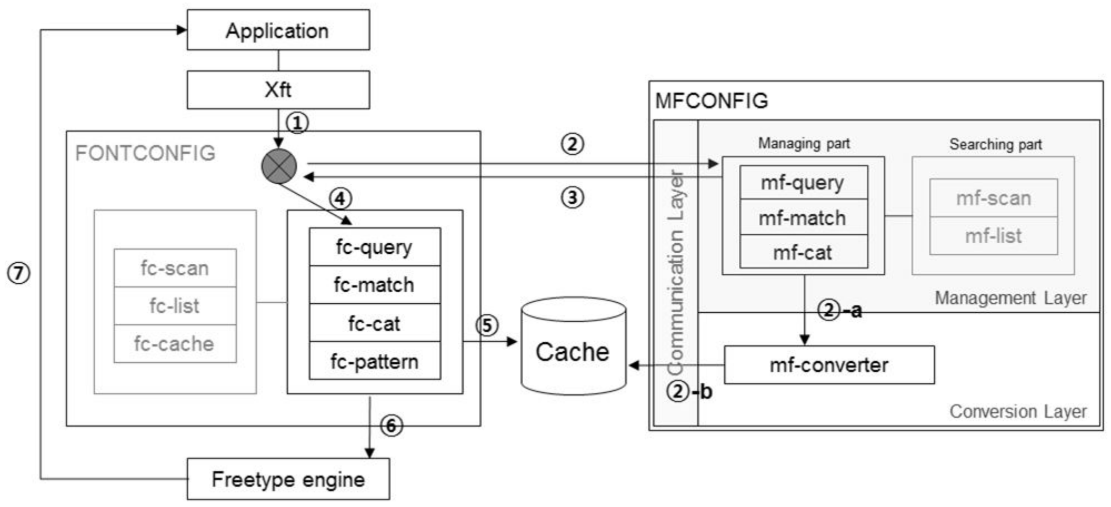


Figure 5: *MFCONFIG* architecture linked fontconfig

The major programs of the *MFCONFIG* module are operated in the Management Layer. This layer is in charge of searching and managing. Searching is an independent function of finding all of the installed METAFONT fonts, and the storing the information in a list beforehand. This list is used for checking whether a specific font is installed or not, and for fetching its information quickly. The searching is implemented with *mf-scan* and *mf-list* that, as shown in Figure 6, work similarly to *fc-scan* and *fc-list* in *FONTCONFIG*, respectively.

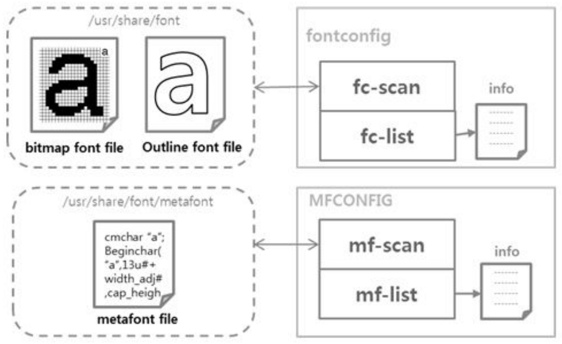


Figure 6: Management layer referenced fontconfig

Management is a core process of the *MFCONFIG* module that is responsible for the following actions: (1) Checking if the requested METAFONT font is prepared in the list, (2) checking if the corresponding outline font is stored with the requested style that is applied to it in the cache memory, and (3) if the outline font is not stored, check whether it needs the conversion of the METAFONT font into the corresponding outline font. If the outline font has already been prepared in the cache, then a notification is sent directly from *MFCONFIG* to*FONTCONFIG* in order to use it. *FONTCONFIG* sends the outline font that is in the cache to the Freetype engine. If the font is not stored in the cache, then the Conversion Layer converts the METAFONT font into the corresponding outline font by applying the style parameters, as shown in Figure 7. The resulting outline font is then stored in the cache, and a notification from the Management Layer through the Communication Layer commands *FONTCONFIG* to use the font.

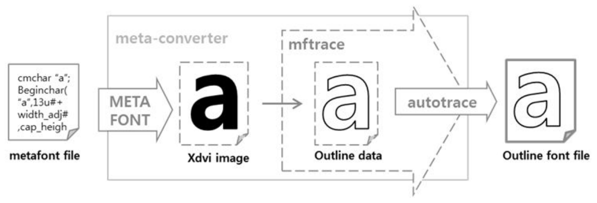
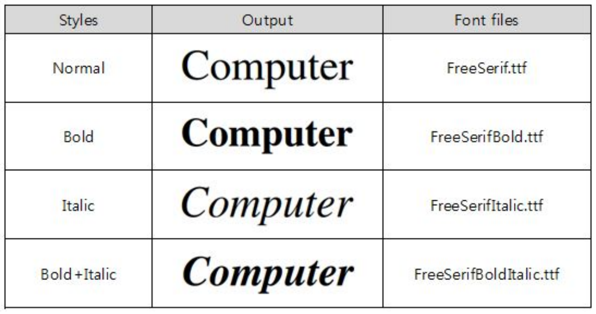


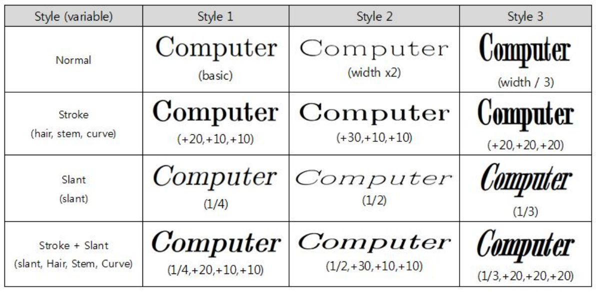
Figure 7: A process of Conversion layer

The work of the *MFCONFIG* module is perfectly compatible with the basic *FONTCONFIG*, and it can append new functions to support of METAFONT. This module works for the management of METAFONT fonts and convert them to the corresponding outline fonts in real-time. When a different style of METAFONT font is requested, *MFCONFIG* can display the resulting font on the screen very conveniently by simply applying the style values to the METAFONT fonts. Therefore *MFCONFIG* has a good usability for METAFONT. In addition, it is not necessary to generate the font-family sets of plain, bold, italic, and italic-bold in advance with respect to *MFCONFIG*, because a variety of font styles can be generated easily by applying the style values.



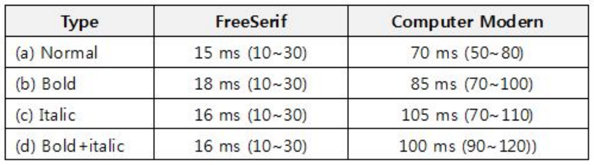
Styles	Output	Font files
Normal	Computer	FreeSerif.ttf
Bold	**Computer**	FreeSerifBold.ttf
Italic	*Computer*	FreeSerifItalic.ttf
Bold+Italic	***Computer***	FreeSerifBoldItalic.ttf

Table 1: Print out 4 files of *'FreeSerif'* font family



Style (variable)	Style 1	Style 2	Style 3
Normal	Computer (basic)	Computer (width x2)	Computer (width / 3)
Stroke (hair, stem, curve)	Computer (+20,+10,+10)	Computer (+30,+10,+10)	Computer (+20,+20,+20)
Slant (slant)	Computer (1/4)	Computer (1/2)	Computer (1/3)
Stroke + Slant (slant, Hair, Stem, Curve)	Computer (1/4,+20,+10,+10)	Computer (1/2,+30,+10,+10)	Computer (1/3,+20,+20,+20)

Table 2: Various style fonts using *'Computer Modern'* typed METAFONT with changing style variable



Type	FreeSerif	Computer Modern
(a) Normal	15 ms (10~30)	70 ms (50~80)
(b) Bold	18 ms (10~30)	85 ms (70~100)
(c) Italic	16 ms (10~30)	105 ms (70~110)
(d) Bold+italic	16 ms (10~30)	100 ms (90~120))

Table 3: Average time for printing out 2 different fonts types (millisecond)

4 Examine *MFCONFIG* module

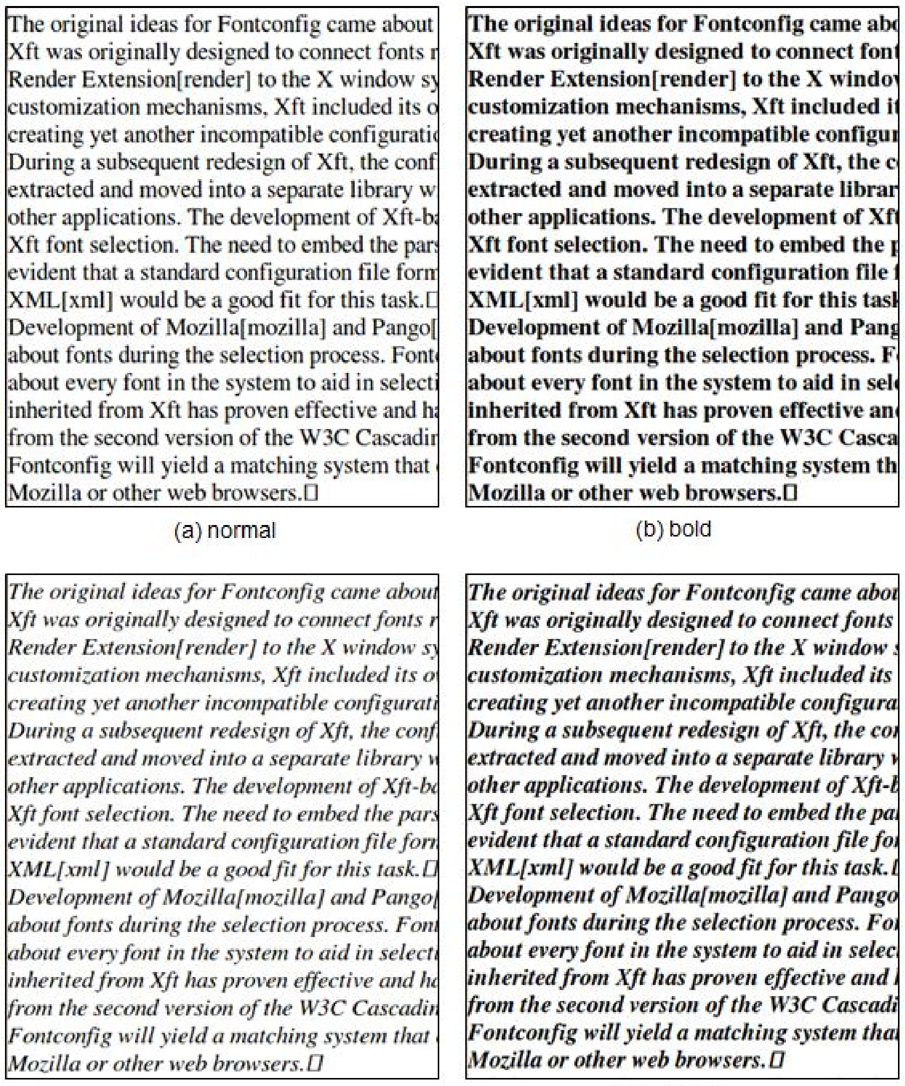
For the performance of the experiments of this stud -y, an application for the use of the X Window system in Linux was developed, and the display of a text file was attempted with the use of a variety of font files. In addition, the TrueType font family named *"FreeSerif"* is used along with the four font styles (*normal, bold, italic*, and *bold+italic*) and a METAFONT font named *"Computer Modern"* were used. The *"FreeSerif"* font family consists of the following four files: *FreeSerif.ttf, FreeSerifItalic.ttf, FreeSerifBod.ttf,* and *FreeSerifBoldItalic.ttf*. Similarly, the *Computer Modern* font was examined along with the four styles *normal, thickness, italic,* and *thickness+italic*. The sample text comprises over 2,000 words and over 8,800 characters, including the space characters. For the performance analysis regarding the Freetype rasterizer, the time between the requesting of a font with styles from an application and the successful display of on-screen text were measured and compared.

Table 1 shows the *"FreeSerif"* font family with the four different styles, and Table 2 shows 12 styles that were generated for the *"Computer Modern"* font of METAFONT. All of these styles were made from one original prototype of the METAFONT font by simple changing of the style parameters. Therefore, the METAFONT font has good capability of generating various font styles with the style values.

For the printing out of a text file on the application, four of the *"FreeSerif"* files from Table 1 and Style 1 of *"Computer Modern"* from Table 2 were used. For Style 1 from Table 2, the four parameter values are *hair, stem, curve, and slant*. The three parameters of *hair*, *stem*, and *curve* are related to the *bold* style, but these parameters are different for lowercase and uppercase. The *slant* parameter is related to the *italic* style. The chosen parameter values for the representation of a bold style are hair+20, stem+10, and curve+10, while the *slant* is 0.25 for a representation of the *italic* style. Figure 8 shows the results of the printing of *"FreeSerif* with four different styles, and Figure 9 shows the results of the printing of the *"Computer Modern"* font.

Table 3 shows the average time to print out of both the *"FreeSerif"* and *"Computer Modern"* contents. In this experiment, the results from 10 ms to 30 ms were obtained, and the average time is 16 ms, while four TrueType fonts were used. Therefore, extra time was required for the conversion of the TrueType fonts. In the case of the *"Computer Modern"* font of METAFONT, the result is much slower than that of *"FreeSerif"*, because it needs additional time for the conversion of the METAFONT font into the corresponding outline font. The obtained results are from 50 ms to 120 ms, and the average time is 90 ms. Even though this time is 10 times slower, 90 ms is still a short time for the printing of a font file on screen. It is possible to conclude that the *MFCONFIG* module could be used with *FONTCONFIG* for the support of METAFONT on a Linux PC almost in real time.

The *MFCONFIG* module is a convenient system to provide users with various styled fonts on screen by applying style parameters directly to the METAFONT font. The users can use METAFONT easily like TrueType font as shown in Figure 9.



The original ideas for Fontconfig came about
Xft was originally designed to connect fonts r
Render Extension[render] to the X window sy
customization mechanisms, Xft included its o
creating yet another incompatible configurati
During a subsequent redesign of Xft, the conf
extracted and moved into a separate library w
other applications. The development of Xft-b
Xft font selection. The need to embed the pars
evident that a standard configuration file form
XML[xml] would be a good fit for this task.□
Development of Mozilla[mozilla] and Pango[
about fonts during the selection process. Font
about every font in the system to aid in selecti
inherited from Xft has proven effective and h
from the second version of the W3C Cascadin
Fontconfig will yield a matching system that
Mozilla or other web browsers.□

(a) normal

The original ideas for Fontconfig came ab
Xft was originally designed to connect fon
Render Extension[render] to the X windo
customization mechanisms, Xft included i
creating yet another incompatible configu
During a subsequent redesign of Xft, the c
extracted and moved into a separate librar
other applications. The development of Xf
Xft font selection. The need to embed the
evident that a standard configuration file
XML[xml] would be a good fit for this tas
Development of Mozilla[mozilla] and Pang
about fonts during the selection process. F
about every font in the system to aid in sel
inherited from Xft has proven effective an
from the second version of the W3C Casca
Fontconfig will yield a matching system th
Mozilla or other web browsers.□

(b) bold

The original ideas for Fontconfig came about
Xft was originally designed to connect fonts r
Render Extension[render] to the X window sy
customization mechanisms, Xft included its o
creating yet another incompatible configurati
During a subsequent redesign of Xft, the conf
extracted and moved into a separate library w
other applications. The development of Xft-bc
Xft font selection. The need to embed the pars
evident that a standard configuration file forn
XML[xml] would be a good fit for this task.□
Development of Mozilla[mozilla] and Pango[
about fonts during the selection process. Font
about every font in the system to aid in selecti
inherited from Xft has proven effective and ho
from the second version of the W3C Cascadin
Fontconfig will yield a matching system that
Mozilla or other web browsers.□

(c) italic

The original ideas for Fontconfig came abo
Xft was originally designed to connect fonts
Render Extension[render] to the X window
customization mechanisms, Xft included its
creating yet another incompatible configura
During a subsequent redesign of Xft, the co
extracted and moved into a separate library
other applications. The development of Xft-
Xft font selection. The need to embed the pa
evident that a standard configuration file fo
XML[xml] would be a good fit for this task.
Development of Mozilla[mozilla] and Pango
about fonts during the selection process. Fo
about every font in the system to aid in selec
inherited from Xft has proven effective and
from the second version of the W3C Cascadi
Fontconfig will yield a matching system tha
Mozilla or other web browsers.□

(d) bold-italic

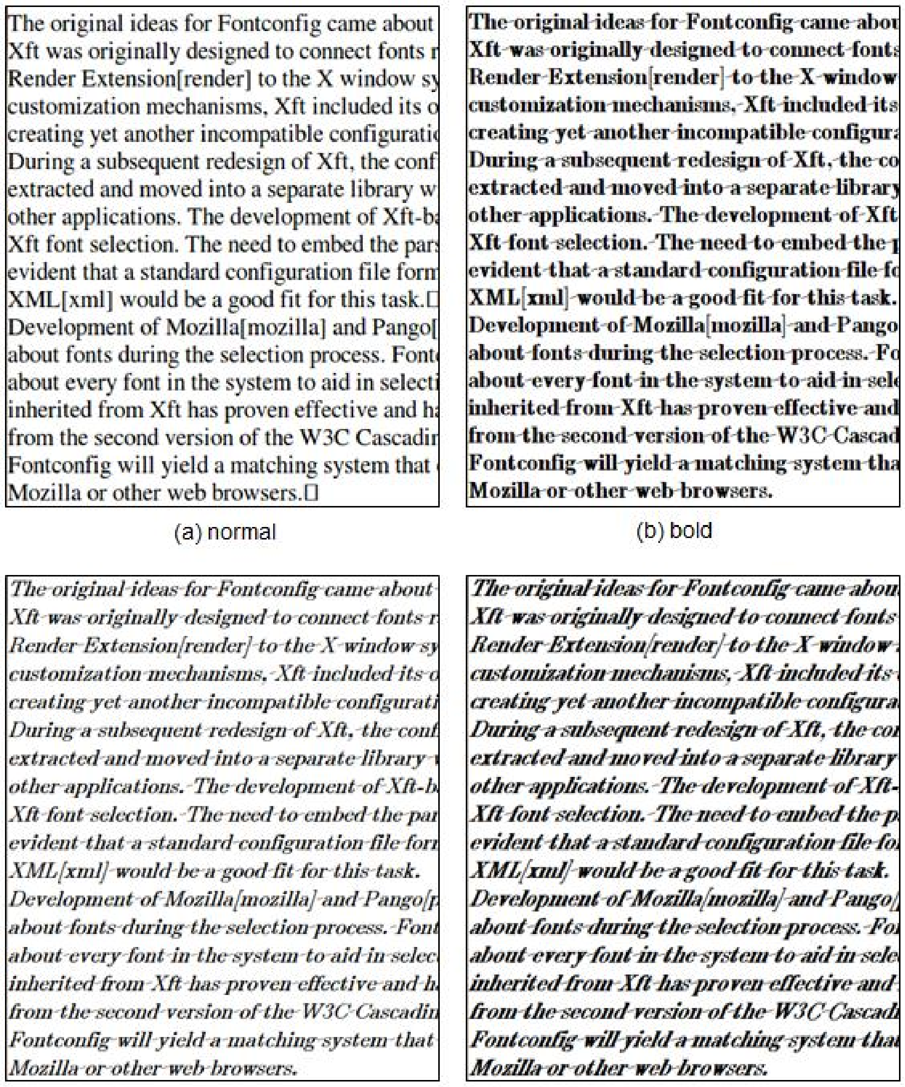
Figure 8: Print out text with four styles of *"FreeSerif"(normal, bold, italic, bold+italic)*

In this paper, the METAFONT font *"Computer-Modern"*, which provides alphanumeric values and symbols, is examined. It is possible to perform tests with the other METAFONT fonts from CTAN (comprehensive TeX archive network) directories that support languages such as Russian and Thai. But difficulty was experienced regarding the testing for which complicated CJK fonts are used.

CJK fonts are very complicated as compared to the alphabet-based fonts and they are composed of several thousands of phonemes. A number of studies have been conducted to partially implement the CJK fonts, such as Hongzi[8][9] and Tsukurimashou[10], including the use of a structural font generator using METAFONT for Korean and Chinese[11], and so on. However there is no commercialized level of files has not yet been created by METAFONT for the CJK fonts. The authors expect that the use of the *MFCONFIG* module for the generation of the CJK fonts will take more time. It may, however, be possible to solve this problem by optimizing meta-converter in the Conversion Layer. Currently, meta-converter works with *"mftrace"* and *"autotrace"* programs, which takes a long time to generate outline fonts.

5 Conclusion

In this paper, the *MFCONFIG* module, which enables the direct use of METAFONT on Linux, is proposed. It is installed and used with the popular Freetype rasterizer. *MFCONFIG* is a plug-in module for the *FONTCONFIG* library of the Freetype engine, and



The original ideas for Fontconfig came about
Xft was originally designed to connect fonts r
Render Extension[render] to the X window sy
customization mechanisms, Xft included its o
creating yet another incompatible configurati
During a subsequent redesign of Xft, the conf
extracted and moved into a separate library w
other applications. The development of Xft-b
Xft font selection. The need to embed the pars
evident that a standard configuration file form
XML[xml] would be a good fit for this task.□
Development of Mozilla[mozilla] and Pango[
about fonts during the selection process. Font
about every font in the system to aid in selecti
inherited from Xft has proven effective and h
from the second version of the W3C Cascadir
Fontconfig will yield a matching system that
Mozilla or other web browsers.□

(a) normal

**The original ideas for Fontconfig came abou
Xft was originally designed to connect fonts
Render Extension[render] to the X window
customization mechanisms, Xft included its
creating yet another incompatible configura
During a subsequent redesign of Xft, the co
extracted and moved into a separate library
other applications. The development of Xft
Xft font selection. The need to embed the p
evident that a standard configuration file fo
XML[xml] would be a good fit for this task.
Development of Mozilla[mozilla] and Pango
about fonts during the selection process. Fo
about every font in the system to aid in sel
inherited from Xft has proven effective and
from the second version of the W3C Cascad
Fontconfig will yield a matching system tha
Mozilla or other web browsers.**

(b) bold

*The original ideas for Fontconfig came about
Xft was originally designed to connect fonts r
Render Extension[render] to the X window sy
customization mechanisms, Xft included its o
creating yet another incompatible configurati
During a subsequent redesign of Xft, the conf
extracted and moved into a separate library w
other applications. The development of Xft-b
Xft font selection. The need to embed the par
evident that a standard configuration file form
XML[xml] would be a good fit for this task.
Development of Mozilla[mozilla] and Pango[p
about fonts during the selection process. Font
about every font in the system to aid in selec
inherited from Xft has proven effective and h
from the second version of the W3C Cascadin
Fontconfig will yield a matching system that
Mozilla or other web browsers.*

(c) italic

***The original ideas for Fontconfig came abou
Xft was originally designed to connect fonts
Render Extension[render] to the X window
customization mechanisms, Xft included its
creating yet another incompatible configura
During a subsequent redesign of Xft, the co
extracted and moved into a separate library
other applications. The development of Xft-
Xft font selection. The need to embed the p
evident that a standard configuration file fo
XML[xml] would be a good fit for this task.
Development of Mozilla[mozilla] and Pango[
about fonts during the selection process. Fo
about every font in the system to aid in sele
inherited from Xft has proven effective and
from the second version of the W3C Cascadi
Fontconfig will yield a matching system tha
Mozilla or other web browsers.***

(d) bold-italic

Figure 9: Print-out text with *"Computer Modern"* font in four styles (*normal, bold, italic, bold+italic*)

makes the latter compatible with the current digital font types of bitmap and outline. The module supports a variety of the styled fonts that are generated from METAFONT along with different parameters.

The existing digital fonts - mainly the outline fonts of Type1 or TrueType - either do not allow users to change their styles, or they can be changed within very limited ranges such as font size. From the experiments of the present study, however, it has been demonstrated that a variety of fonts can be directly generated on screen by applying different style parameters to a prototype of theMETAFONT font on a Freetype rasterizer that is installed with *MFCONFIG*; furthermore, the fonts could be seen within an average time of 90 ms, which is a barely noticeable duration.

MFCONFIG module targets METAFONT fonts to be used with Freetype which is a famous rasterizer. *MFCONFIG* could be used effectively in case of alphabet-based fonts, which are relatively simple and have a limited number of characters. However, there are only a few METAFONT fonts for various languages. It might take a longer time to process CJK METAFONT fonts, which have complicated shapes and have more than several thousands of phonemes. Additional works will be focus on these CJK METAFONT for improving performance, and try to optimize of *MFCONFIG* module. In addition, this module will be experimented such like one of font driver in the Freetype rasterizer.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education(2015R1D1A1A02062211).

References

[1] S. Song. Development of Korea Typography Industry. *Appreciating Korean Language*, 2013.

[2] Y. Park. Current status of hangul in 21th century. *Type and Typography magazine The T*, 7th.

[3] Donald E.Knuth. Computers and typesetting volume c : The METAFONTbook. *TUGboat*, 1986.

[4] K. Packard. Fontconfig. *Gnome User's and Developers European*, 2002.

[5] K. Packard. The xft font library: Architecture and users guide. *XFree86 Technical Conference,Citeseer*, 2001.

[6] H.Kakugawa. Vflib- a general font libray that supports multiple font formants. *EuroTEX confrerence*, March 1998.

[7] H.Kakugawa. A general purpose font module for multilingual application programs. *SP&E*, March 2000.

[8] JR. Laguna. Hong-zi: A chinese metafont. *Communications of the TEX Users Group*, 2005.

[9] Candy L. K. Yiu, Jim Binkley. Qin notation generator. *TUGboat*, 26th, 2005.

[10] Matthew Skala. Tsukurimashou: A japanese-language font meta-family. *TUGboat*, 34th, 2013.

[11] Gyungjae Gwon, Minju Son, Genho Jeoung, Jaeyong Choi. Structural font generator using METAFONT for korean and chinese. 2016 (in preperation).

◇ Jaeyoung Choi
Soongsil University, Seoul, Korea
choi@ssu.ac.kr

◇ Sungmin Kim
Soongsil University, Seoul, Korea
sungmin.kim@ssu.ac.kr

◇ Hojin Lee
Soongsil University, Seoul, Korea
hojini@ssu.ac.kr

◇ Geunho Jeong
Gensol Soft, Seoul, Korea
ghjeong@gensolsoft.com

Astrological charts with `horoscop` and `starfont`

Matthew Skala

Abstract

TeX should create beautiful documents for all fields of human endeavour, and in this talk, I describe one frequently under-served by typesetting systems: astrology. Astrology has its own tradition of written knowledge and symbolic notation, analogous to that of mathematics but arguably even older; and like mathematics, astrology presents unique challenges for typesetting. Writers of astrological software often focus their attention primarily on the calculations, leaving any kind of graphical presentation as an afterthought. In this talk I present the `horoscop` and `starfont` LaTeX packages, meant for creating visually appealing astrological documents using TeX. No prior knowledge of astrology, and not too much of TeX, will be assumed.

1 `starfont` and `horoscop`

Human beings have looked for meaning in the sky since early prehistoric times. Some of our oldest written materials record the phases of the moon, and seasonal variations in the rising and setting of the sun. Astrological goals such as eclipse prediction motivated much of early mathematics; and mathematical developments in turn made possible more complicated astrological investigations. As mathematics developed a written symbolic notation, so did astrology. It is only recently that the two disciplines were considered distinct from each other at all.

So if TeX is the best tool for typesetting mathematics, then shouldn't it also be useful for typesetting astrology? The question is especially important because there are very few other good tools available. Historically, authors of astrological documents would draw their charts by hand, or use hand-set type. The availability of computers has made astrological computations much easier and more precise; but graphical output from astrological software is often disappointing.

I wrote the LaTeX packages called `starfont` and `horoscop`, starting around 2003, to bring high-quality astrological typesetting to the TeX world. Both are available from CTAN.

The `starfont` package provides the two fonts named StarFont Sans and StarFont Serif, designed by Anthony I.P. Owen. These fonts include signs of the zodiac like ♈♉♍ and planet glyphs like ☿♀♂, as well as other astrological and alchemical symbols. Other LaTeX packages offer some of these characters,

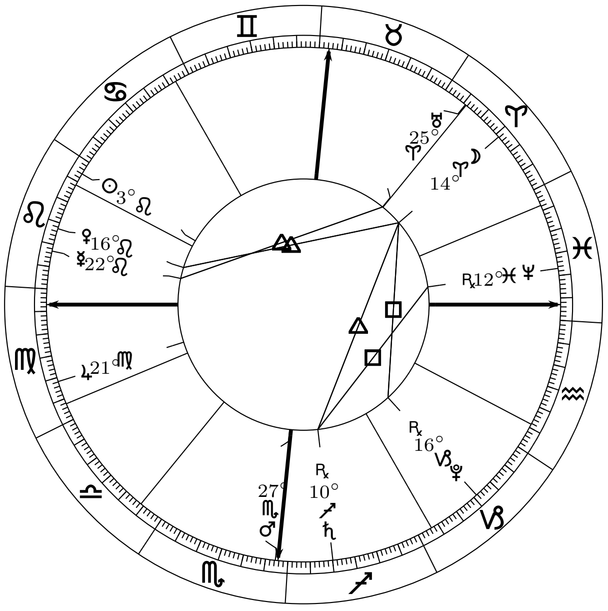


Figure 1: Mundane horoscope for the opening of TUG 2016: 8:45AM, July 25, 2016, Toronto.

but often in ways unsuitable for high-quality astrological typesetting. For example, `wasysym`'s Leo ♌ and North Node ☊ are indistinguishable.

The `horoscop` package's main function is generating wheel charts like Figure 1. This kind of chart represents the sky at a specific time and place in a schematic form that emphasizes the information most relevant to astrological interpretation. The package can take manually-specified coordinates for celestial bodies or interface to external software via `\write18` to calculate their positions.

There are some challenges behind the scenes in typesetting such a chart. Most of the plotting is done in polar coordinates, requiring trigonometric calculations in TeX code. Labels plotted on the chart should not collide even if the objects they represent are near each other in the sky, so the package recalculates positions iteratively, using spring tension. A less visible challenge concerns rounding coordinates to lower precision, because the details of the rounding rules are significant in interpretation.

The `horoscop` package provides a range of chart designs ready-made, and also a framework for users to define their own. It aims to bring TeX's astrology to the same level as TeX's mathematics.

◇ Matthew Skala
Copenhagen
Denmark
`mskala@ansuz.sooke.bc.ca`
`http://ansuz.sooke.bc.ca/`

Development of an e-textbook using LaTeX and PStricks

David M. Tulett, Ph.D [*]
Faculty of Business Administration
Memorial University
St. John's, NL, Canada, A1B 3X5
dtulett@mun.ca

July 25-27, 2016

Abstract

For a course on decision modeling (linear, integer, and goal programming, networks, and decision trees) I created an e-textbook pdf using LaTeX and PStricks. I will discuss why I chose these programs, how I used them, and why I decided to make the final product "open access". The entire pdf can be downloaded from http://stor.mun.ca/handle/123456789/37463.

1 Introduction

A wide-sweeping curriculum change in the undergraduate business programs at Memorial University necessitated the development of many new courses, including Business 2400, Decision Modeling. The introduction of the new courses was staggered, beginning in 2010, with Business 2400 first being offered in Fall Term, 2011. Here is the course description from the 2011-2012 University Calendar:

[*]The author is grateful for funding from a Teaching Fellowship to attend the 2016 TUG Conference.

> **2400 Decision Modeling** provides an introduction to: spreadsheet modeling; linear optimization and the related topics of integer, assignment, and transportation models; and decision analysis including payoff matrices, decision trees, and Bayesian revision. All topics will be taught within the context of business applications. [4]

This new course replaced the former Business 4401 and it differed in the following ways:

1. Business 2400 comes one year earlier in the curriculum than Business 4401 did. Since 4401 (and 4500 Finance) acted as *de-facto* "weed" courses, it was felt that 2400 should come earlier than 4401 did.

2. Business 2400 continued the de-emphasis of the learning of algorithms that began when Business 4401 was created 15 years earlier. The only algorithms taught in Business 2400 are: solving two-variable optimization problems graphically; the graphical method for solving the minimal spanning tree problem; and the rollback procedure for solving decision trees.

3. As the name implies, there is now an increased emphasis on the modeling of problems. In most cases this means defining a set of variables, and then stating an objective function to be maximized or minimized subject to a set of constraints.

4. For the solution of formulated models, only Excel is used. All students at Memorial receive a copy of Microsoft Office, which includes Excel, hence there is no new software to download. Also, learning Excel has to be done anyway, so the only new things would be some specific mathematical functions and the use of the Solver.

When the course was first offered in the Fall of 2011, we adopted a textbook for it: *Managerial Decision Modeling with Spreadsheets* by Render *et al*. [5] At the time, the book sold in the campus bookstore for about $156. (All monetary figures in this article are in Canadian currency.) Alternatively, it could be obtained as an e-book for about $70, but this comes with only a six-month licence.

Even though we had adopted a textbook, before the course had even begun I had begun writing a document with the course title *Decision Modeling*. Principally, this was because the course was about to be offered in Winter 2012, by what was then known as "distance education" (as of 2016, this is now called *online learning*). For the students in this course, my document would substitute for the lectures that they would miss by not being on-campus. Also, I felt that I could improve upon the textbook's coverage of some topics, such as decision analysis (the use of decision trees). Five years later, the *Decision Modeling* document has become an open access stand-alone e-textbook, and the rest of this article describes how this happened.

In section 2, I describe the beginning of the writing: why LaTeX was chosen, the merging of material from earlier courses, and the development of new material. In section 3, I describe what I found useful when I was presented with options for completing various tasks in writing the document. Finally in section 4, I describe why this document became open access.

2 Initial Development

I have been using LaTeX since the early 1990's, so I am very familiar with it and its graphical cousin, PStricks. I am also very familiar with Microsoft Word, it being the standard for word processing where I work. Indeed, I use Word anytime that I need to collaborate with a co-worker, be it for administrative purposes or for writing a journal article. Perhaps if the writing of *Decision Modeling* had been a collaborative effort by multiple professors the document would have had to be written using Word. However, I was doing it on my own, so the choice was mine.

From previous courses I had amassed a large amount of material, mostly written in LaTeX, which could, with modification, be embedded within the new document, partially fulfilling its requirements. For this reason alone it would have made sense to create the *Decision Modeling* document in LaTeX, but there are two other main reasons.

1. LaTeX looks better than Word. This is especially true when creating mathematical expressions.

2. Because things like section numbers are never entered by the user in LaTeX, it becomes much easier to move things about, letting LaTeX figure out how everything (sections, figures, tables, footnotes) is to be renumbered.

All this being said, there is one important development in LaTeX that was needed to make the finished product useful for distribution as an e-document, and that is the creation of **pdfLaTeX**. Back in the 1990's, a .tex file was compiled to create a .dvi file. For anyone with a .tex system, all one had to do was print the .dvi file. However, members of the general public could not be expected to have this ability, and even when stand-alone free .dvi readers came out, it was still a barrier to have to expect people not interested in LaTeX to download the reader. Once the ability to compile a .tex file as a pdf came out, it was a major step forward. While Microsoft Office is a standard piece of software where I work, once a document has been converted to pdf its original source becomes irrelevant, be it Word, WordPerfect, Libre Office Writer, or LaTeX.

Hence, for all these reasons, it made sense to create this document using LaTeX for conversion to pdf. As I have mentioned, Excel is used as the prominent method of solution. For any parts of the document which used Excel, nearly everything had to be written from scratch. In other places I was able to import previously created material, but even here substantial alterations had to be made. The alterations were of two main types:

1. In cutting-and-pasting from several source documents, I was greatly aided by LaTeX's logical design, which frees the user from worrying about the numbering of sections, figures, tables, and footnotes. However, the text had to be extensively edited for phrases such as "as we saw in the previous chapter", which would probably now be an anachronism. Also, there needed to ba consistency about notation, avoiding, for example x_1 in one place while using X_1 in another. More subtle than these things, though, would be the existence of ideas which presume a knowledge which might not apply to readers of the new document.

2. Some of the older documents were based on earlier technology, such as the use of LaTeX's native picture environment (rather than the far more sophisticated PStricks). Over time, these things have been updated.

At the outset of the Fall of 2011, the *Decision Modeling* document consisted of the following chapters, with much content to be completed:

1. **Introduction** Back in 2011, this was a very short chapter, consisting of a look at the paradigm of problem identification, modeling, solution, and implementation; a review of some key Excel functions; and a brief look at professional associations.

2. **Elementary Modeling** This chapter essentially replicated one used in Business 4401, so this required little work.

3. **Applications of Linear Models** Many of the examples used in this chapter had been used in previous courses, but they had never been solved in Excel.

4. **Sensitivity Analysis** Most of the concepts had been seen before, but the pictures to illustrate these concepts had been created in the native LaTeX picture environment rather than PStricks. Also, all computer output needed to be done in Excel.

5. **Network Models** These models are for the assignment, transportation, transshipment, minimal spanning tree, maximal flow, and shortest path problems. The minimal spanning tree problem has a very easy graphically-based algorithm for its solution. For the other five problems, this course stresses the used of Excel, which contrasts with previous courses in which these problems had purpose-built algorithms. Because of this, this chapter needed to be almost written from scratch. Nothing had been written on this chapter as of September, 2011.

6. **Integer Models** Much of the theory had been described in material written for previous courses, but none of the models had been solved in Excel.

7. **Goal Programming and Nonlinear Models** Same problem as the previous chapter.: theory, but no Excel.

8. **Decision Analysis I** Mostly done already, but some Excel work required.

9. **Decision Analysis II** Mostly done already, but some Excel work required.

In addition to all the work that needed to be done to complete all these chapters, another requirement was to create end-of-chapter problems for student completion, and to make the solutions for these problems.

3 Making the Document

The Fall Term proceeded with the Render *et al* textbook, on which all course assignments were based. As a supplement, students in my sections could download a pdf document then called *Business 2400 Decision Modeling Course Manual* (a document title spread over three lines) which had no Chapter 5, and with the problems noted above in the other eight chapters. An immediate first priority was to have something written for Chapter 5. One of these models is the Assignment Problem, for which a small example is presented, along with its mathematical model and its spreadsheet model.

3.1 Assignment Problem Example: Assigning 3 Jobs to 3 Machines

Suppose that we have three jobs, and three machines on which these jobs will be done. Each machine will do just one of the three jobs. All three machines are capable of doing each job, but there are some differences in performance. We can think of these differences in terms of cost (which could be time rather than dollars). Suppose that the costs to assign each job (row) to each machine (column) are as follows:

		Machine		
		1	2	3
	1	30	20	18
Job	2	17	40	21
	3	25	32	28

By inspection the minimal cost solution is to assign job 1 to machine 2, assign job 2 to machine 1, and assign job 3 to machine 3, for a total cost of $20+17+28=65$. To solve this as a mathematical model, we define the meaning of $X_{i,j}$ for all pairs (i,j):

$$X_{i,j} = \left\{ \begin{array}{ll} 1 & \text{if job } i \text{ is assigned to machine } j \\ 0 & \text{otherwise} \end{array} \right\} \quad i = 1,2,3 \quad j = 1,2,3$$

The reason for using the numbers 1 and 0 becomes clear when we write the model. For example, if job 2 is assigned to machine 3 (i.e. $X_{2,3}=1$), then the cost is 21(1) = 21. If job 1 is *not* assigned to machine (i.e. $X_{2,3}=0$), then the cost is 21(0) = 0. Hence, whether or not job 2 is assigned to machine 3, we incur a cost of $21X_{2,3}$.

Hence the objective function is:

minimize $30X_{1,1}+20X_{1,2}+18X_{1,3}+17X_{2,1}+40X_{2,2}+21X_{2,3}+25X_{3,1}+32X_{3,2}+28X_{3,3}$

Every job must be assigned to a machine, hence for each job i one of $X_{i,j}$'s will be 1 (and the other two will be 0), hence the sum will be 1:

$$\begin{array}{rcl} X_{1,1}+X_{1,2}+X_{1,3} & = & 1 \\ X_{2,1}+X_{2,2}+X_{2,3} & = & 1 \\ X_{3,1}+X_{3,2}+X_{3,3} & = & 1 \end{array}$$

Every machine must have a job assigned to it, hence for each machine j one of $X_{i,j}$'s will be 1 (and the other two will be 0), hence the sum will be 1:

$$\begin{aligned} X_{1,1}+X_{2,1}+X_{3,1} &= 1 \\ X_{1,2}+X_{2,2}+X_{3,2} &= 1 \\ X_{1,3}+X_{2,3}+X_{3,3} &= 1 \end{aligned}$$

Finally, each variable must be 0 or 1.

$$\text{all } X_{i,j} \in \{0,1\}\,.$$

In one sense this is a specialized type of linear programming problem, but it seems to violate one of the assumptions of linear programming which requires that all variables be continuous, rather than integer. However, it turns out that the assignment problem is naturally integer. By this, we mean that the solution will only contain 0/1 variables, even when these have not been specifically required. Hence, any software for general linear programming will solve an assignment problem.

We will use the Solver in Excel to solve this type of problem. Indeed, the rectangular array paradigm of Excel is very useful for this type of problem, where the cost data is in this format in the first place.

Since the cost data are in a 3 by 3 array, we can also use a 3 by 3 array for the values of the variables. Note that the SUMPRODUCT function is happy with this; here it's an array times an array on a cell-by-cell basis, not the dot product of one row with another row. In this example it's B3 times B10 plus C3 times C10 and so on up to D3 times D12. This type of product is not the same as matrix multiplication. Here is the setup in formula mode on the spreadsheet, before entering the Solver (the = signs in row 7 and column F are created by typing '=).

	A	B	C	D	E	F	G
1	**Assignment**		Machine				
2	**Problem**	1	2	3	sum		
3	Job 1				=SUM(B3:D3)	=	1
4	Job 2				=SUM(B4:D4)	=	1
5	Job 3				=SUM(B5:D5)	=	1
6	sum	=SUM(B3:B5)	=SUM(C3:C5)	=SUM(D3:D5)			
7		=	=	=			
8		1	1	1			
9							
10	Total Cost	30	20	18			
11	=SUMPRODUCT(B3:D5,B10:D12)	17	40	21			
12		25	32	28			

In the Solver we ask it to minimize A11 by changing variable cells B3:D5, subject to the three constraints B6:D6 = B8:D8, and the three constraints E3:E5 = G3:G5. We click on the "Make unconstrained variables non-negative" box, and ask for the problem to be solved using the "Simplex LP". Solving the model we obtain:

	A	B	C	D	E	F	G
1	**Assignment**		Machine				
2	**Problem**	1	2	3	sum		
3	Job 1	0	1	0	1	=	1
4	Job 2	1	0	0	1	=	1
5	Job 3	0	0	1	1	=	1
6	sum	1	1	1			
7		=	=	=			
8		1	1	1			
9							
10	Total Cost	30	20	18			
11	65	17	40	21			
12		25	32	28			

As we saw earlier, we see from the Solver output that the minimal cost solution is to assign job 1 to machine 2, assign job 2 to machine 1, and job 3 to machine 3, with a total cost of 65.

3.2 Doing the above in LaTeX

The above description uses the **tabular** environment for the cost data, and the **array** environment for the objective function and equations. When it comes to doing the Excel output, it would have been possible to use the tabular environment to mimic a spreadsheet, including the use of colour, but this would have been slow and cumbersome. Instead, I did the file in Excel, highlighted the range that I wanted printed, went to File/Print, set the Printer to Adobe pdf, made the Settings "Print Selection" and "Fit All Columns on One Page", and under Page Setup, then Sheet, then Print, clicked the boxes for "Gridlines" and "Row and column headings". I named this file Assignment2.pdf, then using Adobe Acrobat, this file was cropped and saved. It was imported into the .tex file as:

```
\begin{center}
\includegraphics[scale=0.9]{Assignment2.pdf}
\end{center}
```

This file needed to be re-scaled to 90% of its original size in order not to spill too much into the margin. This re-scaling is a visual trial-and-error process. It wasn't needed for the final numerical workbook:

```
\begin{center}
\includegraphics{Assignment3.pdf}
\end{center}
```

When importing Excel files in this manner, the ribbon with its words (File, Home, Insert, etc.) and all the icons does not appear. Normally, I think that this is to be preferred – it emphasizes that the most important part of a workbook is what lies in the rows and the columns. If including the ribbon is desired, we can use the Print Screen command and then import this file into Adobe Acrobat. In my document, I do this only once at the outset, to show what the ribbon looks like.

With this section on the assignment problem, plus other sections for the other network models, I finally had a first draft of Chapter 5 completed before the outset of the Winter Term, 2012. Next came the inclusion of dozens of pdf's created by cropping files created in Excel for the many problems covered throughout the document. A major piece of work remained until 2015, the creation of the graphical images needed for Chapter 4 (Sensitivity Analysis). The earlier work on which this chapter was based included graphics made in the native LaTeX **picture** environment. While this environment was better than nothing, it was very inadequate. Everything was in black-and-white, with very limited ability to draw curved lines.

Even straight lines were limited to horizontal and vertical lines, and lines whose rise over the run could be expressed as $a:b$, where a and b are integers from 1 to 6 inclusive.

The way to improve the graphics is to use better software, such as PStricks, which like LaTeX can be freely obtained from TUG. A good introduction is provided in Chapter 5 of *The LaTeX Graphics Companion* by Goossens *et al*. [3] A complete description of PStricks is contained in *PStricks: Graphics and PostScript for* TeX *and* LaTeX by Voss. [8]

At the outset of Chapter 4, a problem is stated and solved. Here follows the problem description, model formulation, and graphical solution.

3.3 A Two-Variable Example

3.3.1 Problem Description

Wood Products Limited buys fine hardwoods from around the world from which they make specialized products for the quality furniture market. Two of their products are two types of spindles.

A type 1 spindle requires 6 cuts, then 4 minutes of polishing, followed by 6.5 minutes of varnishing. A type 2 spindle requires 15 cuts, then 4 minutes of polishing, followed by 4.75 minutes of painting. There is one cutting machine which can operate up to 135 cuts per hour. There is one polishing machine – allowing for maintenance it can operate up to 54 minutes per hour. Both the varnish and paint shops can only handle one spindle at a time. Because of a periodic need for high volume ventilation, the varnish and paint shops cannot be operated continuously. These shops are available for production 58.5 and 57 minutes per hour, respectively.

For each type 1 spindle produced, the company obtains a contribution to profit of \$3. For each type 2 spindle produced, the contribution to profit is \$4. How many spindles of each type should be produced each hour so that the total contribution to profit is maximized?

3.3.2 Model

We define:

X_1 — the number of type 1 spindles produced per hour

X_2 — the number of type 2 spindles produced per hour.

For reference, each constraint is identified by a word description on the left-hand side, and by a number in brackets on the right-hand side.

$$\begin{array}{rrcrcrl}
\text{maximize} & 3X_1 & + & 4X_2 & & & \\
\text{subject to} & & & & & & \\
\text{Cutting} & 6X_1 & + & 15X_2 & \leq & 135 & (1) \\
\text{Polishing} & 4X_1 & + & 4X_2 & \leq & 54 & (2) \\
\text{Varnishing} & 6.5X_1 & & & \leq & 58.5 & (3) \\
\text{Painting} & & & 4.75X_2 & \leq & 57 & (4) \\
 & & & & & & \\
 & X_1 & , & X_2 & \geq & 0 &
\end{array}$$

3.3.3 Graphical Solution

Because of the two 4's in the polishing constraint, this constraint will be on a diagonal. Since it's $\leq$, the arrow indicating feasibility will point south-west. So, since $54/4 = 13.5$, having a 14 by 14 grid must contain the optimal solution. Using these boundaries, we obtain:

							First Point	Second Point
Cutting	$6X_1$	$+$	$15X_2$	$\leq$	135	(1)	$(0,9)$	$(14,3.4)$
Polishing	$4X_1$	$+$	$4X_2$	$\leq$	54	(2)	$(0,13.5)$	$(13.5,0)$
Varnishing	$6.5X_1$			$\leq$	58.5	(3)	$X_1 = 9$	horizontal
Painting			$4.75X_2$	$\leq$	57	(4)	$X_2 = 12$	vertical

The graph (displaying both numerical labels and the names of the constraints) is shown in Figure 1.

We see that constraints (1) and (2), i.e. the cutting and polishing constraints, are binding. The equations we need to solve are:

$$\begin{array}{rcl}
6X_1 + 15X_2 & = & 135 \\
4X_1 + 4X_2 & = & 54
\end{array}$$

Multiplying the second equation by 6/4 = 1.5 we obtain:

$$\begin{array}{rcl}
6X_1 + 15X_2 & = & 135 \\
6X_1 + 6X_2 & = & 81
\end{array}$$

Subtracting the bottom from the top gives $9X_2 = 54$, and hence $X_2 = 6$. Therefore $4X_1 + 4(6) = 54$, hence $4X_1 = 30$, and therefore $X_1 = 7.5$. Putting $X_1 = 7.5$ and

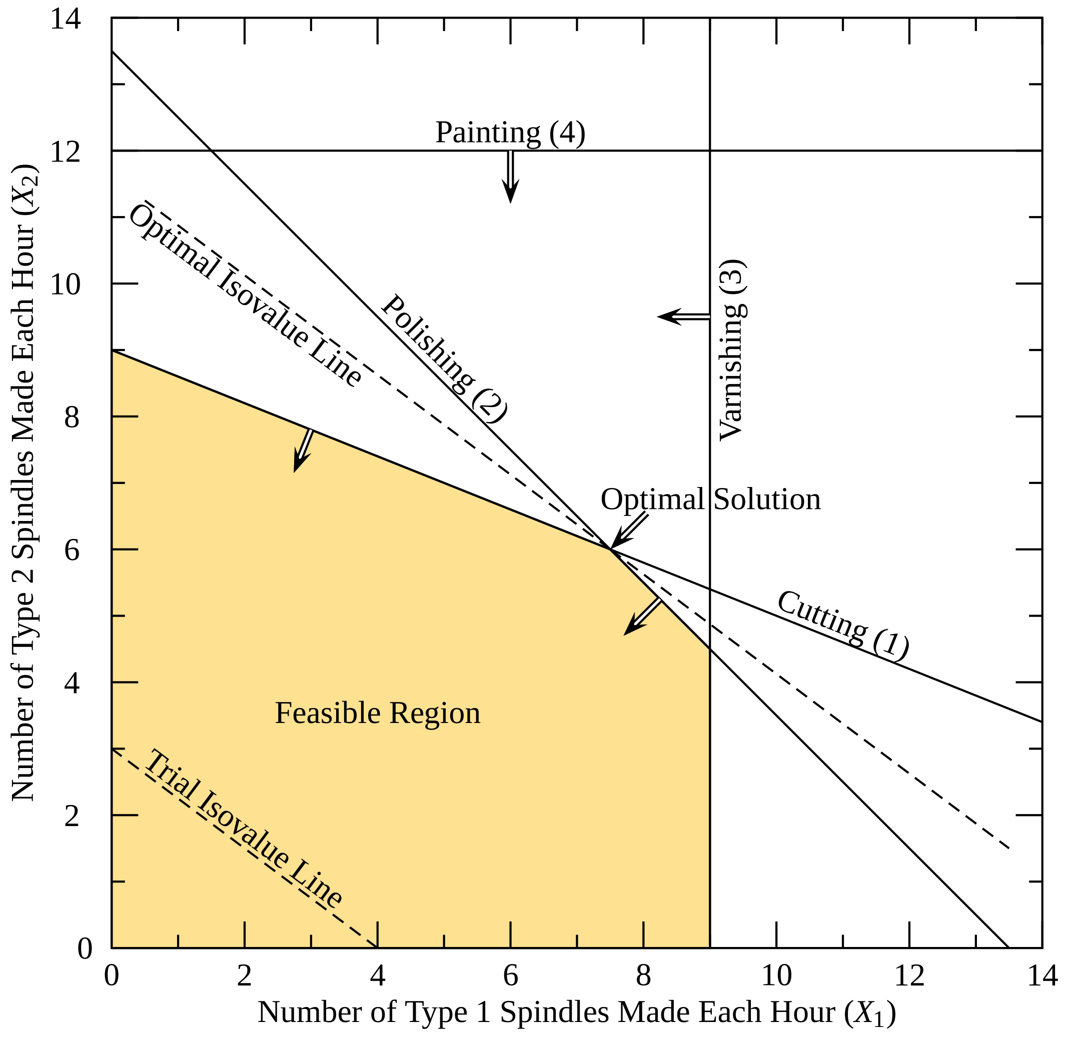


Figure 1: Spindle Problem

$X_2^* = 6.0$ into the objective function we obtain $\text{OFV}^* = 3(7.5) + 4(6) = 46.5$. The 7.5 Type 1 spindles per hour simply means that we must produce 15 of them every two hours, hence the fractional solution is not of concern.

3.4 Doing the Above in PStricks

Everything in the preceding section is easy until we come to the graph. What we need is to be able: to draw straight lines at any angle; to write text next to these lines at the same angle; to draw arrows at right angles to these lines indicating which side is true for the inequality; to fill in the polygon which represents the region in which all inequalities are true. PStricks gives all these things, though a bit of geometry/trigonmetry is needed to make it all work properly. Here are some of the special features of this picture, showing the relevant PStricks code:

1. We have some lines which are neither horizontal nor vertical. If we take the Cutting constraint as an example, we have identified that the boundary of the inequality, which is the line $9X_1 + 15X_2 = 135$, passes through $(0,9)$ (on the vertical axis), and $(14,3.4)$ (on the right-hand side boundary). I found it useful to plot points as integers, so all algebraically determined points were multiplied by 100, making these coordinates (0,900) and (1400,340), but then to use a scaling command in PStricks to make the graph for the page properly. For this graph, I used a quarter scale, using the PStricks command `\psset{unit=0.25pt}`, and then created the line using the PStricks command `\psline(0,900)(1400,340) % (Cutting)`.

2. The set of points for which all inequalities are true is called the *feasible region*. To identify this region, it is highlighted in colour with the built-in paint program in PStricks. The problem is, we need to give PStricks the set of coordinates which gives the vertices ("corners") of the feasible region. This requires doing some successive solving of two equations in two unknowns to determine these points. They are the origin at (0,0); then where the vertical axis meets the boundary of the cutting constraint, which is (0,9); then where boundary of the cutting constraint meets the boundary of the polishing constraint which is (7.5,6); then polishing and varnishing at (9,4.5), then varnishing and the horizontal axis at (9,0). I defined my own colour using the PStricks `\defincolor` command, which I named marygold. (I know that the flower is named *marigold*, but I named this colour after my wife, whose name is Mary.) Therefore the code includes

```
\definecolor{marygold}{cmyk}{0,0.1,0.5,0}
```

and

```
\pspolygon[fillstyle=solid,fillcolor=marygold]
(0,0)(0,900)(750,600)(900,450)(900,0)
```

3. We want to write a label next to each constraint, so that the label is parallel with the constraint. This requires a recall of trigonometry from high school. For the cutting constraint, the inequality is $6X_1 + 15X_2 \leq 135$, hence the rise over the run of the boundary is -6 over 15, or -0.4. To find the angle that the boundary makes with the horizontal axis we need to find the arctangent of this number. PStricks uses degrees, but Excel's ATAN function uses radians, hence we need to use the DEGREES function as well. The Excel command is therefore `=DEGREES(ATAN(-6/15))` which is rounded to -21.801. A related issue is the determination of the coordinates for the centre of the expression "Cutting (1)". For me, this was accomplished by trial-and-error. It isn't just a matter of making the words close but not too close to the line; the words need to be placed so that they don't interfere with other lines or words.

Each diagram done in PStricks requires a lot of work, with many iterations to get everything right. Each iteration requires filename.tex $\Longrightarrow$ dvips$\Longrightarrow$ ps2pdf. Because of all the work for each diagram, I created each diagram with its own file. When I was satisfied with the final pdf version, I cropped it in Adobe Acrobat, and then used an `\includegraphics` command in the main file.

3.5 Continuous Improvement

Until recently, the file, now called DecisionModeling.pdf, was updated at least once every four months. At the outset, references were made to Excel 2010, then Excel 2013, and very recently Excel 2016. That being said, I tried to make all references to Excel to be as version-free as possible, showing the rows and columns without the ribbon.

Other improvements include: adding new examples; adding more problems to the section in each chapter on "Problems for Student Completion"; and making pedagogical improvements to the writing when students needed more explanation about a topic. Any new work is prone to typographical or other kinds of errors, so each revision provides an opportunity to make the necessary corrections.

4 Making the Document Open Access

4.1 Introduction

My university uses a student shell called D2L ("Desire to Learn") which is used for storing documents, submitting assignments, and storing grades. It is used always for online learning courses, and at the professor's option for on-campus courses as well. Access to the course-specific part of D2L is restricted to those with valid course registrations. Hence, a document can be released to the students by posting it to D2L, without making the document publicly accessible.

It was obvious from the outset that this document should be made available to the students free-of-charge. It was, after all, a work-in-progress, rather than a finished product. Also, there would I think be a conflict-of-interest question in a professor adopting his own self-made publication and then collecting royalties from students. Nevertheless, during this period of development, the document contained the words "©David M. Tulett", because I didn't want anyone else to claim my work as their own, and I didn't know what else I could have written in this regard. I did, however, put a note in the course outline to the effect that any student who wanted to have the document printed in a copy shop had my permission to do so.

Once the document had been completed, I had to think about whether I should try to publish it as a commercial venture or let it be freely accessible to anyone who wanted to read it. I chose the latter, for the following reasons:

1. The content of the book was written specifically for the needs of Business 2400 at Memorial University. Other universities might consider a topic such as queueing theory, or simulation, or the study of algorithms, to be essential. Because of this, textbook publishers want manuscripts which are inclusive of all topics in the field, which is why book lengths often reach a thousand pages. To pursue this, I would have to complete much more material.

2. I am unhappy with what has happened to textbook prices. The textbook mentioned earlier now costs about $190. There are plenty of other textbooks costing well over $200. Most students find these prices hard to afford.

3. The existence of free material on the web helps learning.

4. Many prefer to help the public good rather than seek royalties. The TEX community is a good example of this. I can use LATEX and PStricks without paying a fee because of the generosity of those who donated their time and effort to create and maintain these programs. In the software world, this is called "Open Source". There are all sorts of things freely available: alternative suites to Microsoft Office; games such as chess; programming languages, and so on.

4.2 Open Access and Creative Commons Licences

In a conversation with Jeannie Bail, [1] a librarian at the Queen Elizabeth II Library at Memorial University, she informed me that for written material the equivalent of open source is "Open Access". She went on to say that I should read about a major organization acting in the public good in this field, called *Creative Commons* at http://creativecommons.org/. [2] They offer free licences which are described on their website. The most restrictive licence allows anyone to freely download the material but places the following restrictions (quoted verbatim from the Creative Commons website):

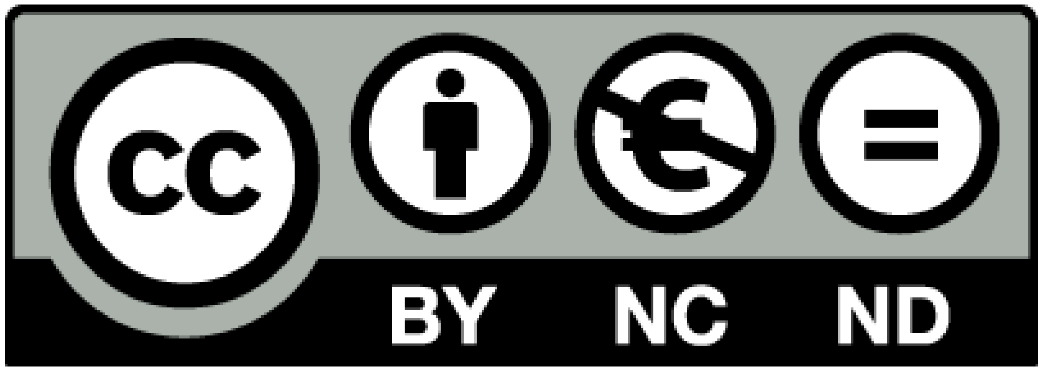
1. Attribution You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

2. NonCommercial You may not use the material for commercial purposes.

3. NoDerivatives If you remix, transform, or build upon the material, you may not distribute the modified material.

The above is a quick summary of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International Licence. The full licence is available at http://creativecommons.org/licenses/by-nc-nd/4.0/.

The reader of DecisionModeling.pdf is informed of this licence by the following graphic:



This image appears in page i, which follows the title page. Also, the word *copy-right* and the copyright symbol © now no longer appear in the document.

4.3 Public Availability

Memorial University encourages open access, as it helps to promote learning, and makes a website available for this purpose. For anyone who simply wants to explore what's there, the address is http://stor.mun.ca/. On the website the purpose of stor is described:

> stor is a learning object repository that aims to promote an atmosphere of sharing where learning objects can be searched, reused, repurposed and contributed. A learning object is a digital, open educational resource that is created to assist in a learning event. [6]

The edition of September 1 2015 was placed online at this site in March 2016. More recently DecisionModeling.pdf has been amended, referencing Excel 2016 instead of Excel 2013. The amended file was uploaded to stor in June, 2016. [7] A direct link to DecisionModeling.pdf is available at:
http://stor.mun.ca/handle/123456789/37463.

With the document now in a repository, I can inform someone of its existence by sending an email message, with a link to the document. Before this, I would have had to attach the file, which at 8 MB is rather large. I now need to make fellow educators in the field of Decision Modeling aware of the document's existence. I thank the TEX community for helping to make this possible.

References

[1] Jeannie Bail, Librarian at the Queen Elizabeth II Library, Memorial University, St. John's, NL, Canada. Personal communication, 2015.

[2] Creative Commons at http://creativecommons.org/, accessed June 2016.

[3] Michel Goossens, Frank Mittelbach, Sebastian Rahtz, Denis Roegel, and Herbert Voss, *The LaTeX Graphics Companion*, 2nd edition, Addison-Wesley, 2008.

[4] Memorial University, 2011-2012 *Calendar*, online at: http://www.mun.ca/regoff/calendar/2011_2012/sectionNo=BUSI-0288

[5] Barry Render, Ralph Stair, Nagraj Balakrishnan, and Brian Smith, *Managerial Decision Modeling with Spreadsheets*, 2nd Canadian Edition, Pearson Canada, Toronto, 2010. ISBN: 978-0-13-208013-2 HD30.25.M35 2010 http://wps.pearsoned.ca/ca_ph_render_mdm_2/.

[6] http://stor.mun.ca/, accessed June 2016.

[7] David Tulett. DecisionModeling.pdf, June 2016. Available at: http://stor.mun.ca/handle/123456789/37463.

[8] Herbert Voss, *PStricks: Graphics and PostScript for* TeX *and* LaTeX, UIT, Cambridge, England, 2011.

Do you need on-site training for LaTeX?

Contact Cheryl Ponchin at

`cponchin@comcast.net`

Training will be customized for your company needs.

Any level, from Beginning to Advanced.

28 years of TeX

In 1988, after 5 years of experience with TeX, Kaveh Bazargan founded River Valley Technologies (originally Focal Image Ltd) in London. River Valley was one of the first companies to deliver TeX services to publishers. The turning point was in 2000 with Kaveh's "structured TeX" coding scheme. The coding is being used to this day, and has allowed us to work with any publisher and with any page style, using virtually identical coding for the TeX files. In addition, the system has allowed us to deliver fully structured XML files directly generated from TeX. We have one of the largest groups of TeX experts in the world with some 500 man years of TeX experience.

As a free service to the TeX community, we are proud to have recorded some 14 TeX conferences on video, including 7 TUG meetings. These are available on **Zeeba.tv**, and many have been viewed more than 10,000 times on YouTube alone.

River Valley will continue to serve the TeX community and to spread the word about TeX and LaTeX.

RIVER VALLEY
TECHNOLOGIES

www.RiverValleyTechnologies.com

Tenth ConTEXt user meeting

Kalenberg, The Netherlands

September 25–October 1, 2016

`meeting.contextgarden.net`

The Porcupine's Quill

PRESERVING THE CULTURE OF THE

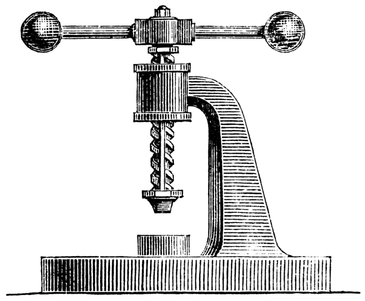
PRINTED WORD

SINCE 1974

❧

We use 20th century printing technology
to make quality books that look and feel like
19th century letterpress products.

porcupinesquill.ca | *facebook.com/theporcupinesquill* | *@porcupinesquill*



Press sharply.

TUG 2016 program

Monday July 25	8:00 am	*registration*	
	8:45 am	Pavneet Arora, Bolton, ON	*Opening: Passport to the TEX canvas*
	9:30 am	Geoffrey Poore, Union Univ.	*Advances in PythonTEX*
	10:00 am	Stefan Kottwitz, Lufthansa Industry Sol.	*TEX in industry I — programming Cisco switches using TEX*
	10:30 am	*break*	
	10:45 am	Stefan Kottwitz	*TEX in industry II — designing converged network solutions*
	11:15 am	Amartyo Banerjee, S.K. Venkatesan, TNQ	*A Telegram bot for printing LATEX files*
	11:45 am	Frank Mittelbach, LATEX3	*Alice goes floating — global optimized pagination including picture placements*
	12:45 pm	*lunch*	
	1:45 pm	Michael Doob, Univ. of Manitoba	*baseball rules summary*
	2:00 pm	Boris Veytsman, George Mason Univ.	*Making ACM LATEX styles*
	2:30 pm	Norbert Preining, Ishikawa, Japan	*Security improvements in the TEX Live Manager and installer*
	3:00 pm	Arthur Reutenauer, Royal Opera House	*The TEX Live M sub-project*
	3:45 pm	*break*	
	4:00 pm	Kevin Larson, Microsoft	*Reading between the lines: Improving comprehension for students*
	≈ 5 pm	*end*	
Tuesday July 26	8:25 am	*announcements*	
	8:30 pm	Kaveh Bazargan, River Valley Technologies, UK	*A graphical user interface for TikZ*
	9:00 am	Matthew Skala, IT Univ. of Copenhagen	*Astrological charts with* `horoscop` *and* `starfont`
	9:30 am	David Tulett, Memorial Univ.	*Development of an e-textbook using LATEX and PStricks*
	10:00 am	Christian Gagné, Univ. Laval	*An Emacs-based writing workflow inspired by TEX and WEB, targeting the Web*
	10:30 am	*break*	
	10:45 am	Arthur Reutenauer, Mojca Miklavec	*Hyphenation past and future:* `hyph-utf8` *and* `patgen`
	11:15 am	Jim Hefferon, Saint Michael's College	*A LATEX reference manual*
	11:45 am	Federico Garcia-De Castro, Alia Musica	*TEXcel?*
	12:15 pm	Jennifer Claudio, Oak Grove High School	*A brief reflection on TEX and end-user needs*
	12:45 pm	*lunch*	with typeforming video (40 min)
	2:00 pm	Jaeyoung Choi, Seoul, Korea	*MFCONFIG: Metafont plug-in for the Freetype rasterizer*
	2:30 pm	Michael Sharpe, UC San Diego	*New font offerings — Cochineal, Nimbus15 and LibertinusT1Math*
	3:00 pm	*break*	
	3:15 pm	Robert Bringhurst, Quadra Island, BC	*The evolution of the Palatino tribe*
	4:15 pm	*TUG meeting*	
	≈ 5:15 pm	*end*	
	5:15 pm	Herbert Schulz, Naperville, IL	*Workshop: TeXShop tips & tricks*
Wednesday July 27	8:25 am	*announcements*	
	8:30 am	Abdelouahad Bayar, Cadi Ayyad Univ.	*Towards an operational (LA)TEX package supporting optical scaling of dynamic mathematical symbols*
	9:00 am	John Plaice, Montreal, QC	*Zebrackets: A score of years and delimiters*
	9:30 am	Charles Bigelow, Bigelow & Holmes	*Probably approximately not quite correct: Revise, repeat*
	10:30 am	*break*	
	10:45 am	David Walden, East Sandwich, MA	*Some notes on the history of digital typography*
	11:15 am	Tim Inkster, The Porcupine's Quill	*The beginning of my career*
	12:15 pm	*lunch*	with road painting video (10 min)
	1:45 pm	*group photo*	
	2:00 pm	Joe Clark, Toronto, ON	*Type and tiles on the TTC*
	3:00 pm	*break*	
	3:15 pm	Jennifer Claudio	*The case for justified text*
	3:45 pm	Boris Veytsman	*Are justification and hyphenation good or bad for the reader?*
	4:15 pm	Charles Bigelow	*Looking for legibility*
	≈ 5:15 pm	*end*	
	6 pm	*Type and Tile Subway Tour, Joe Clark*	typography discussion at 3–5 subway stops.