OpenType math font development: Progress and Challenges

Dr. Ulrik Vieth

One of the main reasons for the development of the LuaTEX engine has been to provide support for Unicode and OpenType font technology, which by extension implies support for Unicode math and OpenType math as well. One important ingredient is the development of full-featured OpenType math fonts, which are needed to replace traditional math font technology. In this paper, we review recent progress in OpenType math font development as well as the many challenges faced by font developers of OpenType math fonts.

1. Introduction

In this paper, we will discuss technical details of OpenType (OpenType) math fonts, so we will assume that readers are somewhat familiar with the basic concepts of Unicode (Unicode) math and OpenType math font technology.

When we speak about Unicode math, we refer to an effort that was undertaken between 1998 and 2002 by a group of scientific and technical publishers to get math symbols and alphabets accepted into Unicode.

As a result of this activity, literally hundreds of math symbols as well as dozens of math alphabets have been added to Unicode, and have become part of the official standard ever since Unicode 3.2 released in 2002 [1, 2]. From a technical point of view, Unicode math is really nothing special, just a convenient term for a subset of Unicode that is relevant for typesetting math.

When we speak about OpenType math, we refer to an extension of the OpenType font format [3] that was developed by Microsoft when they introduced support for math typesetting in MS Office 2007 [4, 5].

As a result of this, a new optional MATH table has been added to the OpenType font format, which is used to store all the additional information needed for proper typesetting of math, such as font metric parameters controlling the spacing of math as well as additional lookup mechanisms of glyph variants [6, 7].

From a technical point of view, OpenType math does represent an extension of the OpenType font format, but it uses a well-defined extension mechanism, so the optional MATH table will only be seen by typesetting engines which happen to know about math, while other engines will safely ignore it.

Finally, it is also helpful to understand how Unicode math, OpenType math, fonts and typesetting engines work together.

Unicode math, by itself, only defines an encoding for mathematical input. It does not define any semantics as to how a math formula is arranged or how it is spaced. That is a matter left to the font technology (OpenType) and the typesetting engine (LuaTEX or XƎTEX).

In Unicode, each math symbol is usually represented only once, regardless of how many sizes may be needed for proper typesetting. The only exception are letters of math alphabets, where each font shape of each letter has separate slots, since a font change in math may also convey a different meaning.

OpenType, as a font technology, provides the glyphs and the metric information for mathematical output. OpenType math fonts are encoded based on Unicode, but can extend beyond the scope of Unicode.

Where Unicode math only defines a single slot for each symbol, OpenType math provides lookup mechanisms for multiple sizes of glyph variants as well as glyph substitutions for constructed symbols.

Where Unicode math does not define any provisions for the semantics of math, OpenType math provides a table to store the font metric information controlling the spacing, but it leaves it to the typesetting engine to interpret and apply these parameters.
In the end, it all depends on having an OpenType math-capable typesetting system to take advantage of the information in OpenType math fonts.
When OpenType math was introduced, MS Office was the only available OpenType math engine, but both XƎTEX and LuaTEX have since implemented OpenType math capabilities [8, 9]. While XƎTEX only provides a partial implementation, LuaTEX aims to provide a full-featured OpenType math engine.

2. Progress in OpenType math font development

When OpenType math was introduced, there was only a single math font available: Cambria Math, which was developed by Tiro Typeworks on behalf of Microsoft and bundled with MS Office 2007.
In some sense, the situation was reminiscent of the early days of TEX, when Computer Modern was the only available math font in MetaFont format.
In recent years, several more OpenType math fonts have been added, so by 2011 there are at least 4 choices of math fonts available as released versions:

- Cambria Math, by Tiro Typeworks on behalf of Microsoft [11],
- Asana Math, by Apostolos Syropouls [12],
- XITS Math, by Khaled Hosny, derived from the STIX fonts [13, 14],
- Latin Modern Math, by the GUST e-foundry [15].

Several more choices of math fonts are under development and will be ready for release sooner or later:

- Neo Euler, by Khaled Hosny on behalf of DANTE, in cooperation with Hermann Zapf [16, 17],
- Lucida Math, by Khaled Hosny on behalf of TUG, in cooperation with Bigelow & Holmes [18],
- Minion Math, by Johannes Küster [19].

Finally, several more math font projects are planned or announced:

- TEX Gyre Math, by the GUST e-foundry,
- Maxwell Math, by Tiro Typeworks [20].

Given all these recent and ongoing font projects, we will soon have OpenType math font support for a number of popular text typefaces, such as Latin Modern, Times, Palatino, Lucida Bright, or Minion.
In some sense, the situation is now reminiscent of the early days of PostScript font support for TEX, when choices of math fonts were still very few, but several popular typefaces were already supported.
It may be interesting to note that most OpenType math fonts were developed by the same few teams or developers: Tiro Typeworks (Cambria, Maxwell), GUST e-foundry (Latin Modern, TEX Gyre), Khaled Hosny (XITS Math, Neo Euler, Lucida Math).
We may conclude that OpenType math font development remains a challenging task, that can be mastered only by a few specialized teams or individuals.

3. What is the basis for OpenType math font development?

Before we consider the challenges faced by font developers of OpenType math fonts, it may be worthwhile to consider the question: What is the basis for OpenType math font development?
First, there is a specification of the OpenType MATH table, developed by the Microsoft typography group. The specification is officially considered experimental and is available to developers only on request, so it has remained

unpublished, despite the fact that it has since been widely adopted as a *de facto* standard by multiple typesetting engines and font tools.

Next, there is a reference implementation of an OpenType math font, Cambria Math, developed by Tiro Typeworks on behalf of Microsoft. This font is widely available, and can be viewed with font editors such as FontForge, making it conveniently possible to study how the font is constructed and what it contains.

Finally, there is a reference implementation of an OpenType math typesetting engine, which happens to be MS Office, developed by the Microsoft Office group. Unlike the specification, which is at least somewhat open, the reference implementation is completely closed and off-limits, so it is hardly possible to verify how the specification is actually implemented.

Given this scenario, developers of OpenType math fonts or engines are facing the problem to determine what really defines the reference behavior and what may be needed to make their fonts behave correctly when used with different typesetting engines.

First, the OpenType math specification may not be perfect, leaving some gray areas open to questions or interpretations. For example, there is hardly any clear description when to apply an italic correction.

Next, the reference OpenType math font may not be perfect either. For example, it may have some incorrect parameter settings, which may confuse some engines when interpreting the parameter values literally.

Finally, the reference OpenType math engine may not be perfect either. For example, it may have interpreted the specification in certain ways, or it may have included some undocumented workarounds to patch certain problems in the reference font.

In LuaTEX, the implementation of the OpenType math engine has followed the specification as much as possible, but in case of doubt, it has chosen to follow the reference implementation of MS Office. OpenType math fonts developed and tested with LuaTEX should therefore work equally well with MS Office, although they may not work quite as well with XƎTEX.

4. Challenges in OpenType math font development

The development of an OpenType math font is challenging for many reasons. Besides the inherent complexity, the size of the project is also a factor. Typical examples of OpenType math fonts may include between 1500 and 2500 symbols or letters, not counting size variants or optical design sizes. Besides technical issues and design issues, achieving some level of completeness is already a significant challenge by itself.

4.1 Completeness of math symbols

Unicode math defines hundreds, if not thousands of math symbols. However, developers of OpenType math fonts can choose what to implement and will typically implement only a subset of the most important symbols and will accept some level of incompleteness.

Most OpenType math fonts will include a common subset, comparable to what was available in traditional TEX math fonts, based on 7-bit or 8-bit encodings, but very few OpenType math fonts will provide the complete set of symbols defined in Unicode math.

At the moment, XITS Math is the most complete of all available OpenType math fonts, because it is based on the STIX fonts [21], which have taken nearly a decade to design and review all the glyphs.

At the other end of the spectrum, Neo Euler is the least complete of all OpenType math fonts, which is understandable given that Euler always had to rely on borrowing symbols from other fonts.

All the other available OpenType math fonts rank somewhere in between these extremes, with each of Asana Math, Lucida Math, and Minion Math providing some ranges of additional symbols that go beyond the scope of Cambria Math. By comparison, Latin Modern Math (Beta) is still far less complete.

One important factor to consider when converting TEX math fonts to OpenType format is that a number of macros need to be replaced by designed symbols. This will include triple dots, double and triple integrals, negated or stacked symbols, arrows with hooks or tails, long arrows,

over- and underbraces.
In the end, the degree of incompleteness that can be tolerated depends on the actual usage. If you are only using basic math, the symbol coverage of all available fonts will be more than enough, but if you need special notations, it may be worthwhile to check the symbol tables of the fonts to ascertain which will provide the required symbols.
Probably the best reference of Unicode math symbols for various OpenType math fonts can be found in the documentation of the `unicode-math` package [22].

4.2 Completeness of math alphabets

Unicode math defines more than a dozen shapes of math alphabets, which includes the following:

- 4 shapes of a serif alphabet (regular, italic, bold, bold italic), each including Latin and Greek,
- 4 shapes of sans-serif (regular, italic, bold, bold italic), some including Latin and Greek,
- 2 shapes of Script or Calligraphic (regular, bold), each including upper- and lowercase,
- 2 shapes of Fraktur or Blackletter (regular, bold), each including upper- and lowercase,
- 1 shape of open face or Blackboard bold (regular), also including upper- and lowercase.

Once again, developers of OpenType math fonts can choose what to implement and will typically implement a common subset of the most important alphabets, but will not necessarily provide all the shapes.
With the exception of Neo Euler, which has only an upright design, most fonts will include at least 4 shapes of the main serif alphabet, but the completeness of other math alphabets may vary considerably.
Some fonts may not include any sans-serif alphabets, some may only include an incomplete range of sans-serif (only Latin, no Greek), some may be missing bold Script and Fraktur, and some may be missing lowercase Script or lowercase and numerals in BBold.
Besides missing some alphabets, some fonts may also provide some additional alphabets, such as an alternative italics, or a different style of Script or Calligraphic. Typically, these alphabets will be accessed via OpenType features using numbered stylistic sets.
Again, in the end, it depends on your actual usage how much incompleteness can be tolerated. If you are typesetting physics, you may well be interested in having a bold sans-serif alphabet for typesetting tensors, but you may need them only a few times in a series of books.
In such cases, you may ask if you actually need Greek for tensors, or if you can do with Latin only. And if you need Greek, you may ask if you really need lowercase, or or if you can do with uppercase only.
Depending on your requirements, your choices of math fonts providing the required alphabets may be limited, or you might be able to avoid the limitations. Finally, you may also consider substituting another font for a certain range of math alphabets.
Taking advantage of range substitutions or stylistic sets depends on some support by macro packages, but such provisions are already provided (at least for LaTeX) by the `unicode-math` and `fontspec` packages (on top of the infrastructure of `luaotfload`) [23, 24, 25, 26].

4.3 Choosing typefaces for math alphabets

Unicode math combines a number of different shapes of math alphabets into a single font, including a matching serif and sans-serif typeface, a Script or Calligraphic, a Fraktur or Blackletter, a Blackboard bold, and even a typewriter design (which we will ignore).
In the case of comprehensive font families, such as Latin Modern or Lucida, the choice of matching typeface designs will be obvious, as there is already a set of serif, sans-serif, and other font shapes that have been designed to be used together.
In other cases, however, choosing a set of matching typeface designs leads to a non-trivial design question: What is the proper choice of

a sans-serif to be combined with a given serif font?
For a Times serif font (as in XITS Math), Helvetica may be an obvious choice of a sans-serif font (although this is debatable), but what should be used with Palatino or Euler? Should the sans-serif be another Zapf design (such as Optima) or could it be something else?
Should the sans-serif be a matching design with similar characteristics or should it be a contrasting design? How much similarity is needed to achieve consistency? How much contrast is needed to make individual letters clearly distinguishable?
Answers to such as questions may not be clear or obvious, but at some point font designers or developers will have to make a choice.
In some cases, such as for Neo Euler or Minion Math, decisions have been deliberately left open, leaving the fonts incomplete without any sans-serif alphabets.
In other cases, such as for Asana Math (derived from `pxfonts` and `cbgreek`), decisions seem to have been taken based on what is available and which sans-serif fonts offer a suitable set of Greek besides Latin.
Besides the choice of sans-serif, similar design decisions may arise for the choice of Script, Calligraphic, Fraktur, or Blackboard Bold designs, although there seems to be considerable agreement among different fonts regarding the typical look of mathematical Script, Calligraphic, or Fraktur. Several different fonts seem to be very similar in the overall style, although each one is still different in the individual design [27].
For Blackboard Bold, however, some very different approaches are favored by different designers.
In some cases, such as Cambria Math or Minon Math, the Blackboard Bold design is derived from an open face version of the main serif font.
In other cases, such as XITS Math, Lucida Math, or Latin Modern Math, the Blackboard Bold is actually a completely different style (typically sans-serif), which may be unrelated to the main sans-serif font.

4.4 Choices of Script or Calligraphic

Design choices of Script alphabets are fairly similar among different math fonts and fall into several groups. One group favors a very embellished style of Script:

XITS Math	$\mathscr{ABCXYZ}\ \mathscr{abcxyz}$
Asana Math	$\mathscr{ABCXYZ}\ \mathscr{abcxyz}$
Lucida Math	$\mathscr{ABCXYZ}\ \mathscr{abcxyz}$

Another group favors a more restrained style of Script:

Cambria Math	$\mathscr{ABCXYZ}\ \mathscr{abcxyz}$
LM Math	$\mathscr{ABCXYZ}$
Neo Euler	$\mathscr{ABCXYZ}$

Finally, several fonts also provide a Calligraphic as an alternative to Script (usually for uppercase only):

XITS Math	$\mathcal{ABCXYZ}$ (StylisticSet=1)
Lucida Math	$\mathcal{ABCXYZ}$ (StylisticSet=4)

4.5 Choices of Fraktur or Blackletter

Design choices of Fraktur alphabets are also similar among different fonts:

XITS Math	$\mathfrak{ABCXYZ}\ \mathfrak{abcxyz}$
Asana Math	$\mathfrak{ABCXYZ}\ \mathfrak{abcxyz}$
Cambria Math	$\mathfrak{ABCXYZ}\ \mathfrak{abcxyz}$
LM Math	$\mathfrak{ABCXYZ}\ \mathfrak{abcxyz}$
Neo Euler	$\mathfrak{ABCXYZ}\ \mathfrak{abcxyz}$

The only exception is Lucida Math, which features a completely different style of Blackletter:

Lucida Math	$\mathfrak{ABCXYZ}\ \mathfrak{abcxyz}$

This may be seen as a example showing that not every Blackletter font is equally well suited for use in math.

4.6 Choices of Blackboard Bold

Design choices of Blackboard Bold alphabets again fall into multiple groups. One group favors a serif design which is derived from the main serif font:

Cambria Math	𝔸𝔹ℂℕ𝕆ℙℚℝ𝕏𝕐ℤ 𝕒𝕓𝕔 𝟘𝟙𝟚
Asana Math	𝔸𝔹ℂℕ𝕆ℙℚℝ𝕏𝕐ℤ 𝕒𝕓𝕔 𝟘𝟙𝟚

Another group favor a sans-serif design which may be unrelated to the main sans-serif font:

XITS Math	𝔸𝔹ℂℕ𝕆ℙℚℝ𝕏𝕐ℤ 𝕒𝕓𝕔 𝟘𝟙𝟚
Lucida Math	𝔸𝔹ℂℕ𝕆ℙℚℝ𝕏𝕐ℤ
LM Math	𝔸𝔹ℂℕ𝕆ℙℚℝ𝕏𝕐ℤ 𝕒𝕓𝕔 𝟘𝟙𝟚

Finally, even the designs of individual letters can vary significantly among different math fonts. It should be noted that some users may have fairly strong preferences regarding such details as to whether the stem or the diagonal of the letter 'N' is double-struck.

4.7 Design issues of math alphabets

Besides some high-level design questions regarding the choice of matching typefaces for math alphabets to be assembled in an OpenType math font, there also some low-level design questions regarding the glyph shapes of individual typefaces.
In particular, we may ask: How should an upright Greek look like and how should a bold sans-serif Greek look like compared to a bold serif Greek?
Unicode math defines a number of math alphabets, many of which are supposed to come with a complete set of Latin and Greek for uppercase and lowercase. This applies to all 4 shapes of the main serif typefaces as well as 2 out of 4 shapes of sans-serif.

4.8 Design of upright Greek alphabets

Unlike Unicode math, traditional TEX math fonts did not provide a complete set of Greek in all shapes.
Whereas uppercase Greek was developed similar to uppercase Latin and came in several different shapes, including serif and sans-serif versions, lowercase Greek was only available in italics and bold italics.
As it turns out, simply creating an upright version of lowercase Greek by removing the slant while reusing the same designs of the italic version (as for Latin Modern Math) does not guarantee good results.
Comparing the results to other designs clearly shows that some letters in the unslanted Greek appear far too unbalanced, in particular for $\gamma, \nu, \pi, \epsilon$.

Latin Modern Math (upright = unslanted)
αβγδεζηθικλμνξοπρςστυφξψωϵϑφϱϖ
αβγδεζηθικλμνξοπρςστυφξψωϵϑφϱϖ

Cambria Math (upright = designed)
αβγδεζηθικλμνξοπρςστυφξψωϵϑϰφϱϖ
αβγδεζηθικλμνξοπρςστυφξψωϵϑϰφϱϖ

Obviously, font projects aiming for good results will need to take glyph design of individual math alphabets seriously, at which point a skilled font designer may be needed in addition to a font developer working on the technical aspects of font assembly.

4.9 Design of sans-serif Greek alphabets

Besides the design of upright Greek letter shapes, the design of sans-serif Greek alphabets may pose another challenge to font developers. Strictly speaking, lowercase Greek letter shapes do not have serifs anyway, so whether a Greek typeface design matches the look of a serif or sans-serif largely depends on matching the typical proportions and the typical stroke thickness of such fonts.
Usually, a sans-serif design exhibits a uniform stroke thickness, whereas a serif design exhibits some contrast between thin or thick strokes, but the actual amount of contrast may vary between different fonts.
For the purposes of typesetting physics, letters from serif and sans-serif alphabets may be used next to each other to distinguish between different types of entities (vectors or tensors) by subtle differences in font shape (serif or sans-serif). If the serif font exhibits a high contrast (as in the case of XITS Math), it is easy to tell apart from a sans-serif font, but if the serif font has a fairly

uniform thickness itself (as in the case of Lucida Math), it becomes difficult to tell which one is which.

Lucida Math
αβγδεζηθικλμνξοπρςστυφξψωϵϑϰφϱϖ
αβγδεζηθικλμνξοπρςστυφξψωϵϑϰφϱϖ
XITS Math
αβγδεζηθικλμνξοπρςστυφξψωϵϑϰφϱϖ
αβγδεζηθικλμνξοπρςστυφξψωϵϑϰφϱϖ

Depending on the characteristics of the font, design of a clearly distinct sans-serif Greek alphabet may depend on more factors than just stroke thickness and may also require further adjustments to glyph shapes.

4.10 Technical issues regarding font metrics

Finally, among all the challenges faced by developers of OpenType math fonts, besides achieving completeness and finding solutions to various design issues, there also remain a few technical issues to consider.
Most notably, there is a fundamental difference how glyph metrics are represented in OpenType math fonts (as opposed to traditional TeX math fonts), and how those glyph metrics are interpreted in OpenType math engines that follow the reference behavior of MS Office, such as LuaTeX (as opposed to XeTeX).
In traditional TeX fonts, the actual glyph width used to be represented by the combination of the nominal width and the italic correction, but in OpenType math fonts, the italic correction is disregarded, and only the nominal width is taken into account.
When converting traditional TeX math fonts to OpenType, it becomes necessary to adjust the metrics to match the interpretation in OpenType math engines to ensure proper rendering in LuaTeX and MS Office, while sacrificing the rendering in XeTeX.
In recent font developments, several fonts besides Cambria Math have adopted the same interpretation, including Lucida Math and XITS Math, while others such as Latin Modern still need to be adjusted.

5. Conclusion

In this paper, we have reviewed recent progress in OpenType math font development as well as the many challenges faced by font developers of OpenType math fonts, including completeness of math symbols and math alphabets, design issues, and technical issues.
While significant progress has been made in recent years, resulting in the upcoming release of several font projects, math font development remains challenging and much work remains to be done.

6. References

[1] *Unicode and math, a combination whose time has come: Finally!,* Barbara Beeton, *TUGboat*, 21(3), 174–185, 2000, http://tug.org/TUGboat/tb21-3/tb68beet.pdf

[2] *Unicode Support for Mathematics*, Barbara Beeton, Asmus Freytag, Murray Sargent, Unicode Technical Report UTR#25. 2001, http://www.unicode.org/reports/tr25/

[3] *OpenType specification, version 1.6, 2009,* Microsoft Typography, http://www.microsoft.com/typography/otspec/

[4] *High-quality editing and display of mathematical text in Office 2007,* Murray Sargent, http://blogs.msdn.com/murrays/archive/2006/09/13/752206.aspx

[5] *Mathematical Typesetting: Mathematical and scientific typesetting solutions from Microsoft*, John Hudson, Ross Mills, Promotional Booklet, Microsoft, 2006, http://www.tiro.com/projects/

[6] *Do we need a Cork math font encoding?,* Ulrik Vieth, *TUGboat*, 29(3), 426–434, 2008, Reprinted in *MAPS*, 38, 3–11, 2009, http://tug.org/TUGboat/tb29-3/tb93vieth.pdf, http://www.ntg.nl/maps/38/02.pdf

[7] *OpenType Math Illuminated*, Ulrik Vieth, *TUGboat*, 30(1), 22-31, 2009, Reprinted in *MAPS*, 38, 12–21, 2009, http://tug.org/TUGboat/tb30-1/tb94vieth.pdf, http://www.ntg.nl/maps/38/03.pdf

[8] *XeTeX Live*, Jonathan Kew, *TUGboat*, 29(1), 151–156, 2008, http://tug.org/TUGboat/tb29-1/tb91kew.pdf

[9] *Math in LuaTeX 0.40*, Taco Hoekwater, *MAPS*, 38, 22–31, 2009, http://www.ntg.nl/maps/38/04.pdf

[10] *Unicode Math in ConTeXt*, Hans Hagen, *MAPS*, 38, 32–46, 2009, http://www.ntg.nl/maps/38/05.pdf

[11] *Projects – Cambria Math*, Tiro Typeworks, http://tiro.com/projects.html

[12] *Asana Math Font*, Apostolos Syropoulos, http://www.ctan.org/pkg/asana-math

[13] *XITS Fonts*, Khaled Hosny, http://www.ctan.org/pkg/xits, http://github.com/khaledhosny/xits-math

[14] *STIX Fonts*, STIX Consortium, http://www.ctan.org/pkg/stix, http://www.stixfonts.org/

[15] *e-foundry*, GUST, http://www.gust.org.pl/projects/e-foundry

[16] *Neo Euler Font*, Khaled Hosny, http://github.com/khaledhosny/euler-otf

[17] *Reshaping Euler: A collaboration with Hermann Zapf*, Hans Hagen, Taco Hoekwater, Volker RW Schaa, *TUGboat*, 29(3), 283–287, 2008, http://tug.org/TUGboat/tb29-2/tb92hagen-euler.pdf

[18] *Another incarnation of Lucida: Towards Lucida OpenType*, Ulrik Vieth, Mojca Miklavec, *TUGboat*, 32(2), 169–176, 2011, https://tug.org/members/TUGboat/tb32-2/tb101vieth.pdf

[19] *Minion Math 1.020*, Johannes Küster, http://typoma.de/en/fonts.html

[20] *Fonts – Maxwell Math*, Tiro Typeworks, http://tiro.com/fonts.html

[21] *The STIX Project :From Unicode to fonts*, Barbara Beeton, *TUGboat*, 28(3), 299–304, 2007, http://tug.org/TUGboat/tb28-3/tb90beet.pdf

[22] *Symbols defined by* `unicode-math`, Will Robertson, http://mirror.ctan.org/macros/latex/contrib/unicode-math/unimath-symbols.pdf

[23] *Unicode mathematics in LaTeX: Advantages and Challenges*, Will Robertson, *TUGboat*, 31(2), 211–220, 2010, http://tug.org/TUGboat/tb31-2/tb98robertson.pdf

[24] *The* `unicode-math` *macro package*, Will Robertson, http://www.ctan.org/pkg/unicode-math, http://github.com/wspr/unicode-math

[25] *The* `fontspec` *macro package*, Will Robertson, http://www.ctan.org/pkg/fontspec, http://github.com/wspr/fontspec

[26] *The* `luaotfload` *macro package*, Khaled Hosny et al., http://www.ctan.org/pkg/luaotfload, http://github.com/khaledhosny/luaotfload

[27] *Math alphabets and the* `mathalpha` *package*, Michael Sharpe, *TUGboat*, 32(2), 164–168, 2011, https://tug.org/members/TUGboat/tb32-2/tb101sharpe.pdf