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

Abdelouahad Bayar

Abstract

In processing of digital documents containing mathematical formulas, the handling of dynamic mathematical symbols is still a difficult problem. A tool to compose mathematics should support the typing of variable-sized symbols taking care of optical scaling and supporting the quality of metal typesetting. Until now, there is no tool that allows these possibilities in a direct and operational way.

This contribution describes and puts into practice the basic steps to develop a (LA)TEX package directly based on a parameterized PostScript Type 3 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 while taking into account optical scaling. Formatting (LA)TEX documents using this package is achieved without requirements of special environments or external programs.

We find that the concept of using parameterized 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, e.g., logo graphics.

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

1 The problem

1.1 Class of mathematical symbols

Mathematical formulas are written using static and/or variable-sized symbols. When using a font at a given size, the dimension and shape of a static symbol remain unchanged in the whole 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 another in the same document. As an example, we can cite the width of the hat symbols indicating angles: $\widehat{A}$ and $\widehat{AOB}$. The managing of variable-sized symbols, which we will sometimes call dynamic mathematical symbols,[1] is still a significant challenge in the area of document processing (see later).

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

1.2 Optical scaling

This is a concept used to handle different sizes of the same font. It is in contrast to linear scaling. To produce, say, size 48 using linear scaling, size 12 is merely magnified four times. This is not the case with optical scaling; rather, the building of the character is done by taking into account the eye of the reader. More details on the optical scaling concept are found in [3, 8, 9].

When we consider humans or trees, for example, we can see obviously that they do not grow according to a linear model. The human eye is naturally attuned to the point of view represented by art. 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 [7] will not happen.)

1.3 Metal vs. digital typesetting and optical scaling

In old books, especially those in mathematics, mathematical formulas were typeset using optical scaling. We give an example of a formula taken from [11] 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 achieve in metal typesetting since symbols were processed in their final sizes. However, the support of optical scaling is not easy in automatic systems when computing symbols or fonts, especially in the case of digital typesetting. More information on this point is in [3, 16].

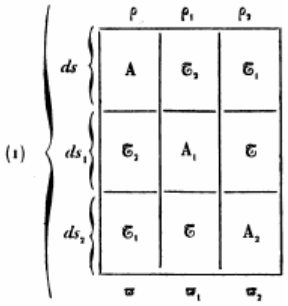

Figure 1: Braces in metal typesetting

1.4 Existing work

Dynamic characters, especially dynamic mathematical symbols, are omnipresent in scientific documents. So, a good application to process scientific documents must include all required means to support dynamic characteristics. In the last four decades, a number of tools have been developed which support

dynamism in different ways. We can cite in this case native TEX [12] and LATEX [13] to support mathematical variable-sized symbols. One important tool to indicate is CurExt [14], 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 one to write Arabic text taking into account the kashida concept. We have to note that the kashida makes manifest an important phenomenon of dynamism in Arabic text typesetting. Detailed information about the kashida and justification of Arabic texts can be found in [10]. It is very important to consider the work accomplished in [2, 3]. This consisted of the design of the "math-fly" font, a PostScript Type 3 font, to supply dynamic mathematical symbols taking into account optical scaling. The particular property of this work, in comparison with the preceding, is that it is used neither under nor with (LA)TEX.

CurExt as a package to extend TEX's capabilities in handling variable-sized symbols has some difficulties in processing mathematical formulas containing more than two (matched) dynamic symbols [14]. So, it does not offer adequate support to manage the general case of mathematical formulas.

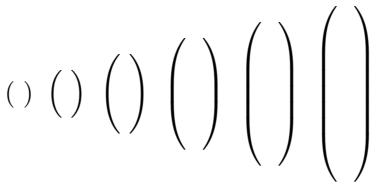

Figure 2: Stretches of parentheses in TEX

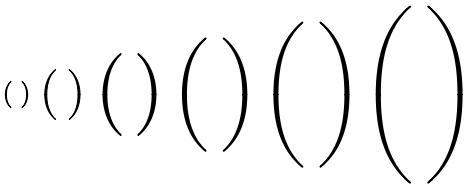

Figure 3: Stretches of parentheses via our dynamic PostScript

As for TEX, readers here likely already know that it supports the composition of mathematical formulas with multiple variable-sized symbols. Optical scaling is not very well supported, however, since the thickness cannot be changed after a certain point (see Figure 2), while the present package can do so continuously (Figure 3). Furthermore, due to the non-dynamic properties of Metafont, dynamic variable-sized symbols at big sizes do not change in shape (Figures 2 and 4), but merely have straight parts extended. The result does not look like the traditional typesetting shown in Figure 1. Figure 5 shows the results for braces with the present package, with the shape changing in a natural way, in addition to the increasing thickness.

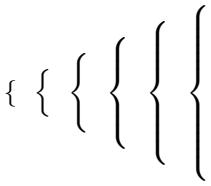

Figure 4: Stretches of left braces in TEX

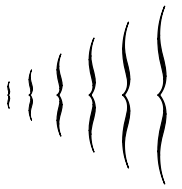

Figure 5: Stretches of left braces via our dynamic PostScript

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

2 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 part concerns defining a *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 in producing good mathematical formulas in a 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 printing. However, in dynamic fonts, characters take their printing characteristics at printing time. More details and examples for comparison are found in [1]. A suitable font to deal with variable-sized symbols must be dynamic; moreover, it must have the following features:

- The language to implement programs encoding dynamic symbols must provide more flexibility in parameterizing symbols. Also, it must have the ability to receive values from outside 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 the Metafont language. This is achieved principally via tfm files. Nevertheless, Metafont does not allow manipulating dynamism at printing time. (LA)TEX can also use other kinds of fonts like PostScript Type 1, TrueType, or the hybrid of these two, OpenType. These fonts are referenced by TEX as if they were virtually Metafont fonts. So the use of these fonts does not add to either a means 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 supports only a semi-dynamism or a discrete dynamism.

PostScript Type 3 fonts have some particular features:

- They use the 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 means 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.

Using PostScript Type 3 does lose 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 may outweigh the disadvantages. Also, nowadays, the computers inside printers are so fast and so large that the time required to move the paper inside printers dominates the printing speed. Moreover, the printer industry regularly makes significant improvements in resolution. Then, the efficiency benefits such as caching and hints are gone.

PostScript Type 3 fonts can be used by TEX in the same way as Type 1. In this case, we cannot take advantage of the benefit of dynamic specification in PostScript. In [2], the authors stated that a parameterized Type 3 font could not be fully used by formatters (editors) such as TEX or others without modifications to the way they call formula symbols. For this reason, they chose to check their font in the Grif project. In our case, the PostScript Type 3 font will be inserted directly into the (LA)TEX source of the document to be formatted by means of the `\special` macro. Of course, the way to process dynamic mathematical symbols in mathematical formulas will be reviewed. The details of the concept are given in the following.

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 Type 3 font is based on the existing possibilities of interaction between (LA)TEX and PostScript. This is done using the command `\special` via the `dvips` driver to translate dvi files to PostScript ones [15]. More precisely, we use these methods to include literal PostScript in TEX documents to work with the PostScript Type 3 font. A summarized design of the package is given below.

- The PostScript Type 3 font supporting dynamic mathematical symbols and all useful procedures is defined in the package as a literal header, a '`!`' `\special`. This is mandatory since the font will be used later when including other PostScript codes to show PostScript dynamic symbols. Our Type 3 font is named "`dynMath`". The command is:
  `\special{! ...dynMath specification...}.`

  The font implements the mathematical symbols which will support curvilinear stretching depending on the values of two global variables

h and w.[2] In the font, the stretching model allows symbols to stretch in height (depth) and width depending on the values of h and w while keeping the same thickness. We note that the scaling is not linear. It is a semi-optical scaling since the thickness is not affected. It is done in macros we will define, such as `\meLeft` for example (see next).

- The principal macro for handling mathematical formulas and thus 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 the dynamic mathematical symbols which are delimiters. Other dynamic symbols (e.g., the radical sign), are defined in separate macros or in some cases existing macros will be redefined. About the delimiters, we chose to finalize at the moment the implementation of only two symbols, namely parentheses and braces. In reality, these 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 notice that variable-sized symbols that are simple combinations of lines are simple to implement in the font.
- In the `\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` determines the stretching amounts of h and w. Then, the corresponding size `fs` at which the font `dynMath` will be used to output the left symbol is calculated.
  3. The dimensions of the left symbol `symHeight`, `symWidth`, and `symDepth` taking into account the PostScript font `fs` are determined.
  4. In an `\hbox` of dimensions `symHeight`, `symWidth`, `symDepth`, the left symbol is written using 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 use simple names like h and w to make the description of the equations easy in this paper. In actual code, meaningful names will be used; for example, h and w will be replaced by `verticalStretch` and `horizontalStretch` respectively.

3.2 The design of the `dynMath` font

The font `dynMath` is based on the font cmex10.mf. The simple difference is that a symbol appears in `dynMath` only once as opposed to the multiple versions in 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 one parameterized 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 at the moment.

We used the existing font cmr10.mf to get the nuclei of left and right parentheses which we parameterized 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 parenthesis in text. For the left and right braces, we saw that none of the Metafont fonts supplied with TEX distributions provide braces that look like the metal ones. So, we designed them. The command applied to generate the basic encoding of the parenthesis through cmr10.mf:

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

To explain the global concepts for designing the font, we use the parenthesis as an example. The same process is applied to the other symbols. Without loss of generality, we show only the top part of the left parenthesis (with respect to the mathematical axis).

Applying MetaPost to cmr10.mf, we get the encoding of the left parenthesis. Its top left part is shown in Figure 6. It is defined based on two Bézier curves linked by two line segments at the top and bottom. To get a parenthesis symbol that is able to stretch when needed, the two Bézier curves are multi-decomposed using the generalized algorithm of refinement [4].

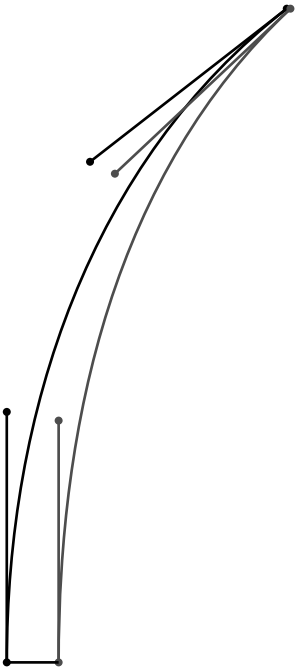

Figure 6: Top part of left parenthesis — initial encoding at size 500

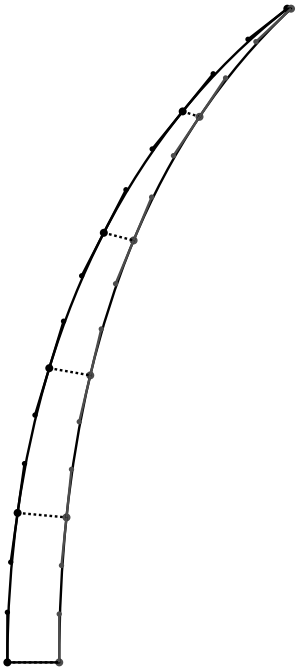

Figure 7: Decomposed top part of left parenthesis (parameterized curves) at size 500 without stretching

To explain in detail, 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 sub-curves. The latter are then parameterized taking 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 at size 500. h and w take the values 250 and 83.33 PostScript points respectively ($83.33 \approx 250/3$). We notice that the initial and stretched versions of the half symbol differ in height and width but they have the same thickness. To emphasize 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 corresponding segment in Figure 8 are parallel and have the same length. It is obvious that the scaling is not linear. It does not support a true 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 the TEX package. The way to parameterize the control points in order to support this semi-optical scaling requires obedience to a strict mathematical model. (We do not present the mathematical concepts here because this is outside the objective of this paper. Nevertheless, we cite the basics of the development.) The curves are parameterized 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 was 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 such as parentheses and braces. This cannot 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 relevant to this paper). Figures 9 and 10 show the two cases of mathematical formulas, namely when the formula is high or deep, respectively. Moreover, they introduce some characteristic variables of formulas:

- f_h: height of formula from the baseline.
- f_d: depth of formula from the baseline.

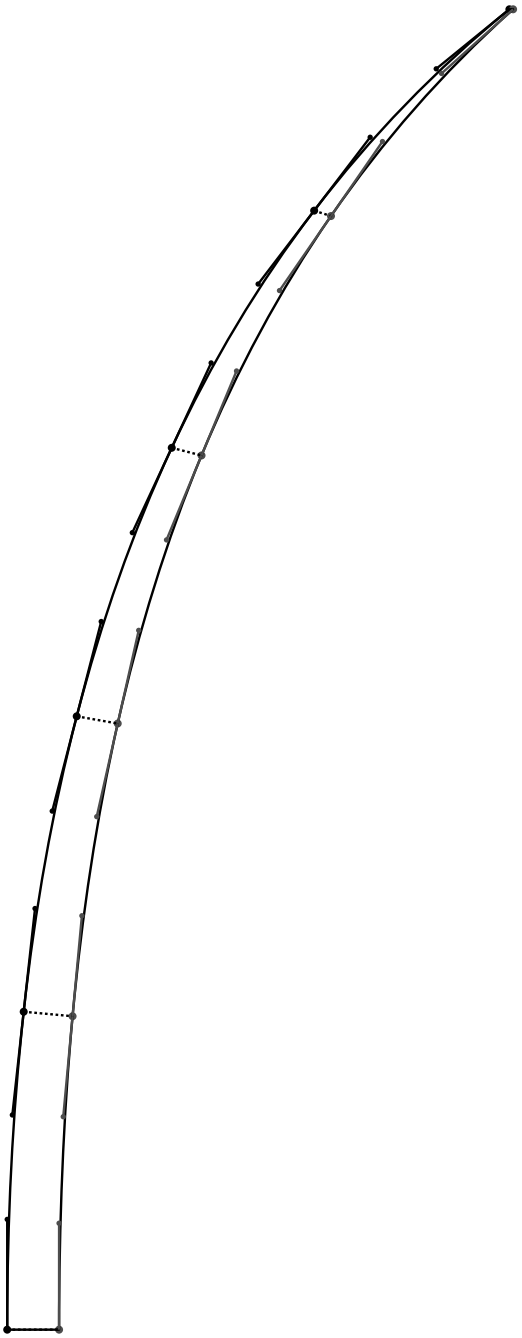

Figure 8: Decomposed top part of left parenthesis (parameterized curves) at size 500, stretched 250 vertically and 83.33 horizontally

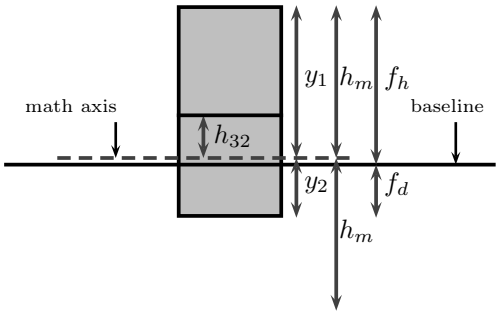

Figure 9: Abstract high mathematical formula

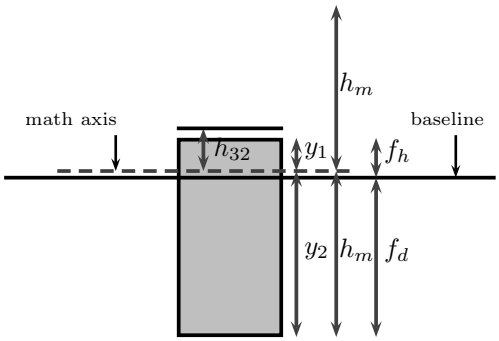

Figure 10: Abstract deep mathematical formula

- y_1: mathematical height of the formula, measured from the mathematical axis to the top of the formula.
- y_2: mathematical depth of the formula, 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 at size 32 (corresponding to the height h_{32}^p in the 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 units corresponding to v_n^p in PostScript units. For example, h_{32} is the height in pt corresponding to h_{32}^p being the height in PostScript of the dynamic symbol at size 32. We have formally $v_n = 1.00375 \times v_n^p$.

One part of optical scaling, as previously stated, is supported by the PostScript Type 3 font whereas the other part is a direct job of the TEX package. After a study of the fonts generated from cmr10.mf and cmex10.mf via MetaPost, we noticed that the standalone parentheses symbols (the symbol consisting of one character) obtained from cmex10.mf are

in some way a linear scaling of the standalone parentheses supplied by cmr10.mf. The dimensions of the big parenthesis in cmex10.mf is approximately three times the size of the ones relative to the parenthesis in cmr10.mf. Consequently, the optical scaling is handled in two different ways depending on the value of h_m. The first case deals with the values of h_m less than or equal to h_{32} whereas the second takes care of values greater strictly than h_{32}.

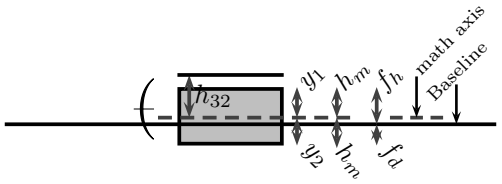

Figure 11: Abstract mathematical formula with $h_m \leq h_{32}$ and non-aligned math axis

When the mathematical height h_m is less than or equal to 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 half of the left parenthesis (taken as an example) equals h_m. Then the font `dynMath` is set to *fs* and the delimiter is written in a `\special`.

We can see that when the delimiter is introduced, the mathematical axis of the formula and that of the delimiter are not aligned. This is normal because the "TEX size" in which the formula (10, 11, ...) is written will usually be different from the delimiter size. In the `\special` macro, a shifting operation is accomplished before writing the PostScript delimiter, as shown here in Figure 12:

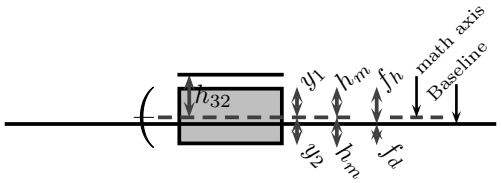

Figure 12: Abstract mathematical formula with $h_m \leq h_{32}$ and aligned math axis

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 to us has a maximal vertical stretching of 32700 bp. As the size 32 is a threshold to handle the symbol stretching, we consider the maximal amount allowed in stretching the half of a parenthesis to be h^p_{max}. The superscript p relates to the PostScript context. We have:

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

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

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

In the following, we give needed dimensions only in TEX points since all calculations are done by TEX. The first step in processing is to determine the size that will be used to compute the delimiter. The PostScript font `dynMath` provides the possibility to extend symbols without affecting the thickness. This is what remains to support 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 be delimited. This is given in Equation 3:

$$e\,(h_m) = c_1 h_m + c_0 \qquad (3)$$

where c_1 and c_0 are constants satisfying the following requirements:

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

For λ, e_{1000} and e_{32} known, we can determine the size *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 Equations 4 and 5.

$$fs = \frac{1003.75}{e_{1000}} e \qquad (4)$$

$$h_{fs} = \frac{h_{1000}}{1003.75} fs \qquad (5)$$

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

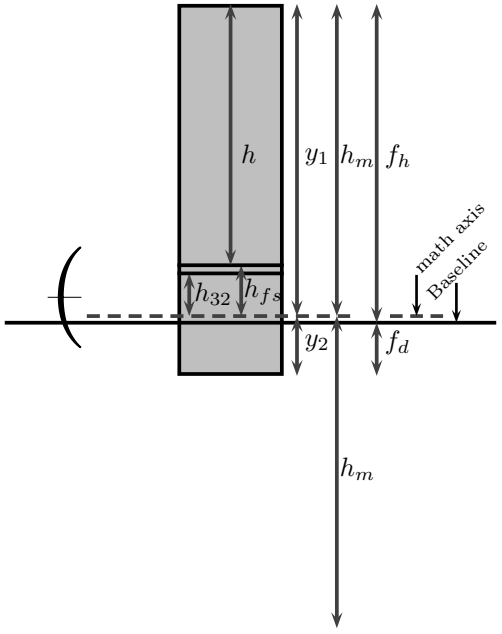

Figure 13: Abstract mathematical formula with $h_m > h_{32}$, non-aligned math axis and non-stretched parenthesis

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 letting $h = h_m - h_{fs}$. (See Figure 13 above.)

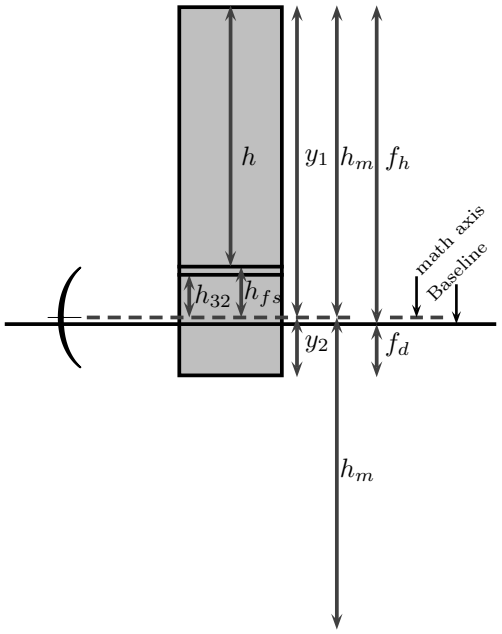

Figure 14: Abstract mathematical formula with $h_m > h_{32}$, aligned math axis and non-stretched parenthesis

The dynamic symbol is written, especially the left parenthesis as shown in Figure 13, in `\special` macros using the font fs^p ($fs^p = 0.9962 \times fs$) as the size. We remark that for the case where $h \leq h_{32}$ that the math axis of the formula and that of the parenthesis 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 at the fs size is stretched by h vertically and by $w = h/3$ horizontally.

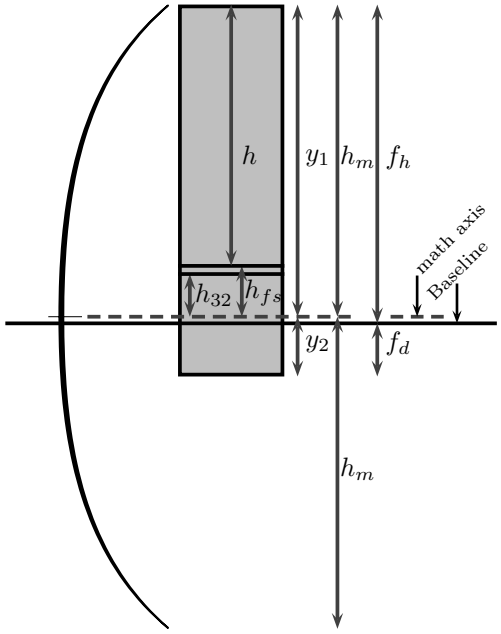

Figure 15: Abstract mathematical formula with $h_m > h_{32}$, aligned math axis and stretched parenthesis

4 Implementation

As we said, the system to typeset mathematical formulas 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 by LATEX. So the package operates with both TEX and LATEX. At the same time, it is a good nucleus from which to develop a LATEX package. One of the important steps in the process of executing 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 all TEX programs. The determination of the current math style imposes the use of complete recursion, i.e., recursion in macro definitions and execution. So the package is very slow and more demanding of memory. To solve this problem, we resorted to the use of Lua(LA)TEX since they supply `\mathstyle` (`\luatexmathstyle`) which permits recognition of the mathematical style on the fly, greatly reducing the time of processing and the use of memory. Of course, since a PostScript Type 3 font is used in conjunction with the TEX package, then source documents are formatted via `dviluatex` (`dvilualatex`) and `dvips` commands.

5 Conclusions

We have developed a mini-package for TEX offering support of variable-sized mathematical symbols with respect to optical scaling. As an illustration, we implemented only two symbols: the parenthesis and the brace. But the support of these two symbols demonstrates the feasibility as well as the possibility to produce scientific documents with the quality of metal typesetting. In the future, we will finish, at both the font and package levels, the support of all the rest of the dynamic mathematical symbols. At the same time, options relating to the printing quality will be added to the final package. A more important task will concern the optical scaling with consideration of an artistic view point. Indeed, the relationship between the values of horizontal, vertical stretching and thickness will be studied taking into account artistic satisfaction.

References

[1] Jacques André, B. Borghi, "Dynamic Fonts", *PostScript Language Journal*, Vol. 2, no. 3, pp. 4–6, 1990. http://jacques-andre.fr/japublis/fontesdyn.pdf

[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. http://jacques-andre.fr/japublis/opticalscaling.pdf

[4] Brian A. Barsky, *Arbitrary Subdivision of Bézier Curves*, Technical Report UCB.CSD 85/265, Computer Science Division, University of California, 1985. http://www.eecs.berkeley.edu/Pubs/TechRpts/1986/CSD-86-265.pdf

[5] M.J.E. Benatia, M. Elyaakoubi, A. Lazrek, "Arabic Text Justification", TUG 2006 conference, Marrakesh, Morocco. *TUGboat* 27:2, pp. 137–146, 2006. http://tug.org/TUGboat/tb27-2/tb87benatia.pdf

[6] Daniel M. Berry, "Stretching Letter and Slanted-Baseline Formatting for Arabic, Hebrew and Persian with ditroff/ffortid and Dynamic PostScript Fonts", *Software—Practice and Experience*, vol. 29, no. 15, pp. 1417–1457, 1999. https://cs.uwaterloo.ca/~dberry/FTP_SITE/tech.reports/keshide.paper.pdf

[7] Harry Carter. "The Optical Scale in Typefounding", *Typography*, No. 4, pp. 2–6, Autumn, 1937. https://issuu.com/letterror/docs/harry_carter_optical_scale_in_typefounding

[8] Circuitous Root, "From the Optical Scale to Optical Scaling", 2016. http://www.circuitousroot.com/artifice/letters/press/typemaking/mats/optical/index.html

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

[10] Mohamed Elyaakoubi, Azzeddine Lazrek, "Justify just or just justify", Journal of Electronic Publishing, Volume 13, Number 1, 2010. http://dx.doi.org/10.3998/3336451.0013.105

[11] G. Lamé, "Leçons sur les coordonnées curvilignes et leurs diverses applications", Imprimerie de Mallet Bachelier, Paris – Rue du Jardinet 12, 1859. https://archive.org/details/bub_gb_jfgxNexuunYC

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

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

[14] Azzeddine Lazrek, "CurExt, Typesetting variable-sized curved symbols", EuroTEX 2003 conference, Brest, France. *TUGboat* 24:3, pp. 323–327, 2003. http://tug.org/TUGboat/tb24-3/lazrek.pdf

[15] Tomas Rokicki, "Dvips: A DVI-to-PostScript translator", version 5.996, 2016. http://tug.org/dvips

[16] 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.

◇ Abdelouahad Bayar
Cadi Ayyad University
Ecole Supérieure de Technologie de Safi
[High college of technology of Safi]
Morocco
a.bayar (at) uca.ma