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]

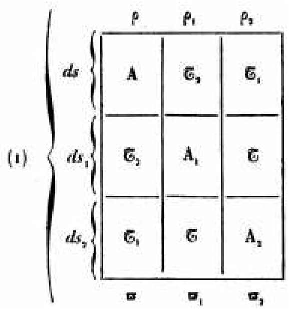

Figure 1: Braces in metal typesetting

[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.

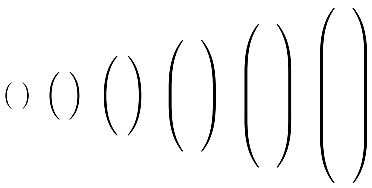

Figure 2: Stretches of parentheses in TEX

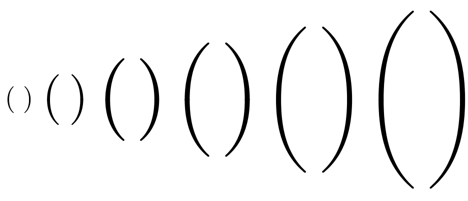

Figure 3: Stretches of parentheses via PostScript

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

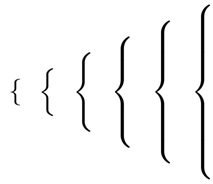

Figure 4: Stretches of left brace in TEX

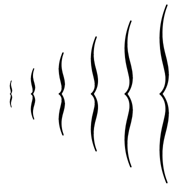

Figure 5: Stretches of left brace via PostScript

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.

preliminary draft, June 26, 2016 0:16

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.

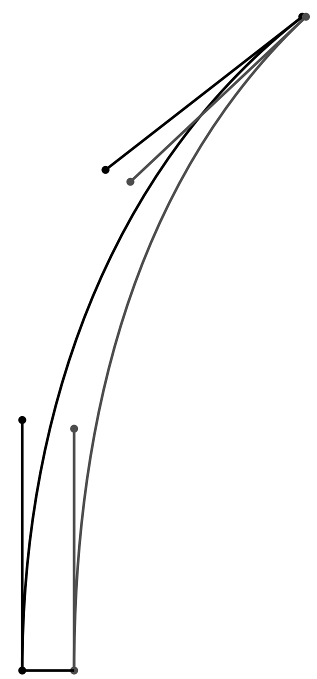

Figure 6: Top Part of left parenthesis - Initial encoding in body 500

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).

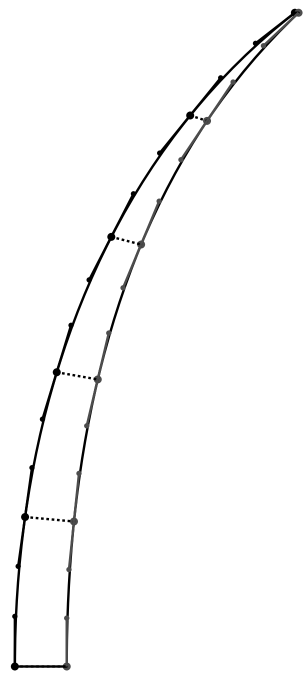

Figure 7: Decomposed Top Part of left parenthesis (Parametrized curves) in body 500 without stretching

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

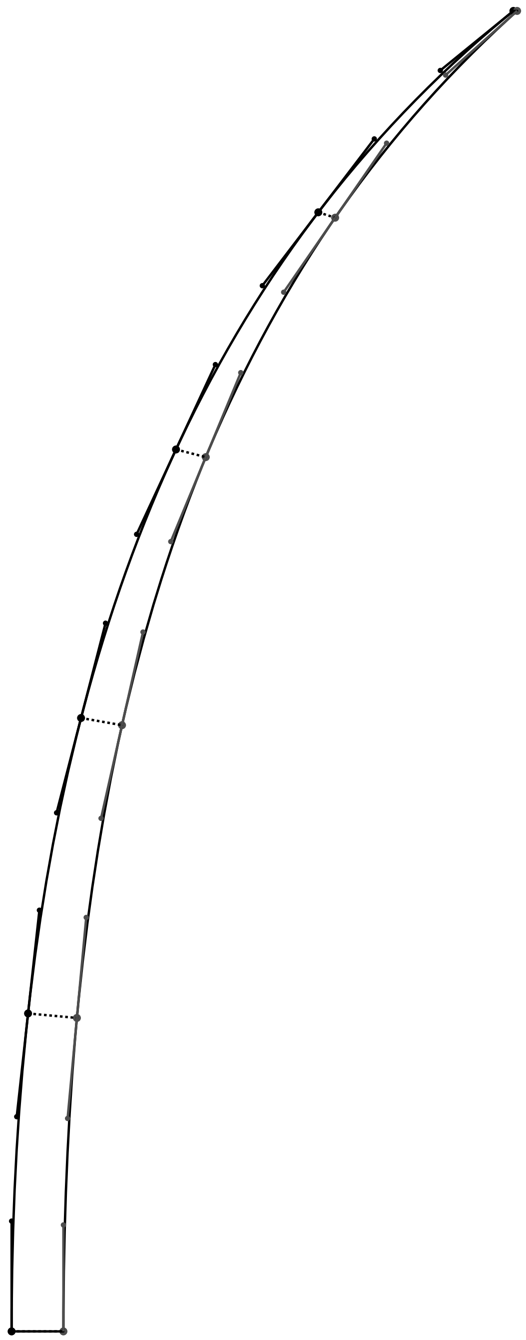

Figure 8: Decomposed Top Part of left parenthesis (Parametrized curves) in body 500 stretched 250 vertically and 83.33 horizontally

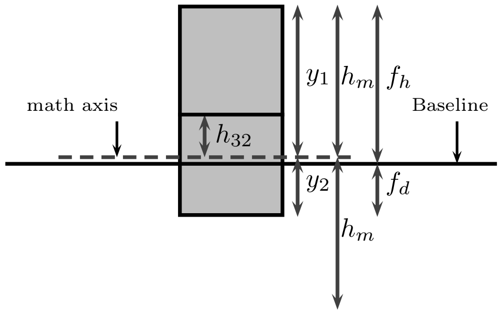

Figure 9: Abstract high mathematical formula

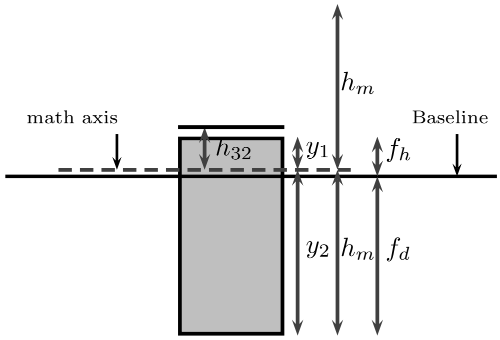

Figure 10: Abstract deep mathematical formula

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_{32}^p 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_n^p in PostScript unit. For example, h_{32} is the hight in pt corresponding to h_{32}^p being the hight in PostScript of the dynamic symbol in body 32. We have formally $v_n = 1.00375 \times v_n^p$.

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}.

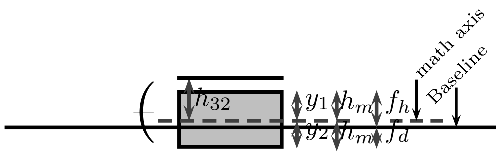

Figure 11: Abstract mathematical formula with $h_m \leq h_{32}$ and non aligned math axis

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.

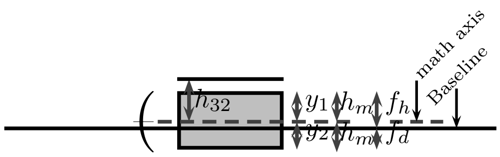

Figure 12: Abstract mathematical formula with $h_m \leq h_{32}$ and aligned math axis

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:

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

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 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.

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

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.

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

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

[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

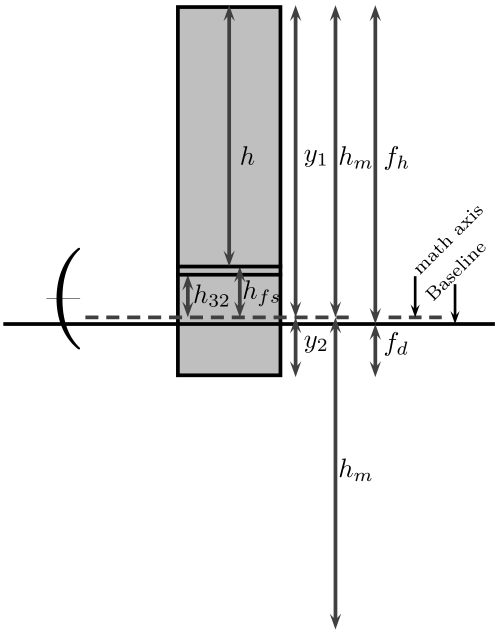

Figure 13: Abstract mathematical formula with $h_m > h_{32}$, non aligned math axis and non stretched parenthesis

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

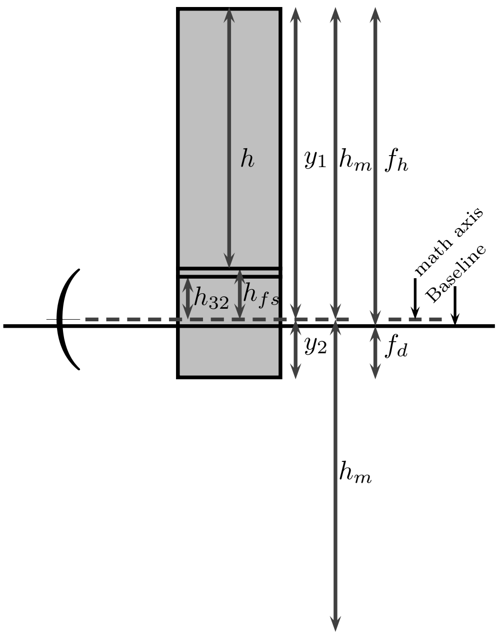

Figure 14: Abstract mathematical formula with $h_m > h_{32}$, aligned math axis and non stretched parenthesis

The dynamic symbol is written, especially the left parenthesis as showed in Figure 13, in `\special` macros using the font fs^p ($\mathit{fs}^p = 0.9962 \times \mathit{fs}$) 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.

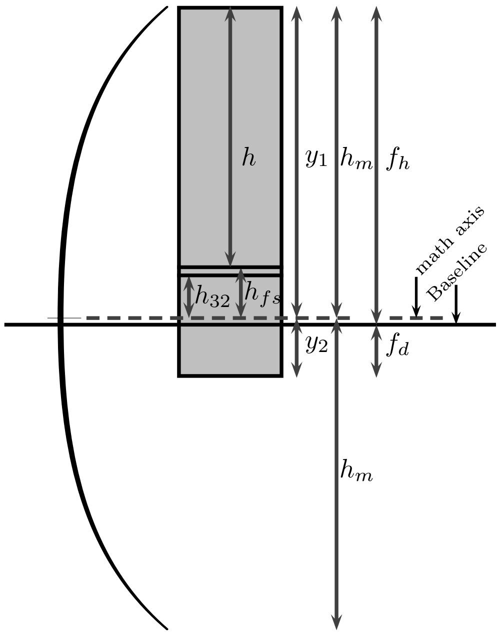

Figure 15: Abstract mathematical formula with $h_m > h_{32}$, aligned math axis and stretched parenthesis

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

 preliminary draft, June 26, 2016 0:16

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.

 preliminary draft, June 26, 2016 0:16