dynMath: Underlying principles of the design

Abdelouahad Bayar

Abstract

This article presents the basic principles underlying the design and development of dynMath, a package that supports dynamic mathematical symbols. The focus is on the interaction between LaTeX and PostScript via the TeX `\special` primitive, and in particular the direct use of a dynamic PostScript Type 3 font in the LaTeX source.

1 Introduction

Electronic documents, especially scientific ones, are typeset using static and/or dynamic characters. The mathematical formula is always the most suitable example for highlighting the subject. Mathematical variable-sized symbols, such as delimiters (parentheses, braces, radicals, etc.), are a good way to make the subject concrete.

When we talk about scientific document processing, we think first and foremost of (LA)TeX in its various implementations: TeX [5], LaTeX [6], LuaTeX [7], etc. Dynamic symbols such as delimiters and others are supported by (LA)TeX but in some cases the properties of *optical scaling*, *uniformity* of shape, *right-sizing* and *metal likeness* are not respected. The dynMath package [4] has been developed with the aim of supporting such characteristics and thus enhancing and improving the typesetting quality of (LA)TeX.

(LA)TeX offers the possibility of interacting with PostScript [1] via the TeX `\special` primitive. The latter makes it possible to insert and manipulate PostScript code in (LA)TeX through the dvips driver [8] while generating PostScript from the dvi files. We have used this mechanism to handle a dynamic Type 3 [1] font in TeX, thus enabling dynamic mathematical symbols to be supported by the dynMath system. The way in which this approach of interaction is used is unusual in the development of (LA)TeX packages. For this reason, we believe it will be interesting to present details of the implementation process. We note that the same work was done when the development of dynMath was launched in 2016 [2]. The resumption of such work is justified by the change that has taken place in the implementation.

This paper is organized as follows. In Section 2, the overall layout of the dynMath system is given. In Section 3, some details of dynMath in terms of the Type 3 font are presented. In Section 4, the way in which dynMath supports dynamic mathematical symbols in terms of TeX programming and interaction with the dynamic font Type 3 is studied. The paper ends with some conclusions and perspectives.

2 dynMath: The layout

The dynMath system is now in its basic state. It contains the minimum necessary files to operate, namely dynMath.sty and dynMath.tps.

- dynMath.sty: this is the LaTeX package itself. It contains the definition of the macros required to support the mathematical variable-sized symbols.
- dynMath.tps: this is the specification of a PostScript Type 3 font parameterized to draw mathematical symbols with dimensions and shapes satisfying given contexts.

Some details of the two files will be seen below to give an idea on how they work. A part highlights the interaction between LaTeX and PostScript Type 3.

3 dynMath: The font

3.1 PostScript inside LaTeX

The requirements for supporting dynamic mathematical symbols are identified in [2]. The PostScript language and PostScript Type 3 fonts are recognized as suitable to provide a solution.

Natively, (LA)TeX has an interface to fonts specified in METAFONT. This is achieved through tfm files. These communicate information about the dimensions, in a definitive static way, of the characters which will appear in the document to be printed. METAFONT is a compiled language and does not allow for manipulating the characteristics of characters at printing time.

It is also possible to use a Type 3 font as a virtual font. Even if a Type 3 font fully uses the PostScript language and can be parameterized in a flexible way, it will not be able to offer support for dynamism via virtual fonts because the latter are seen by TeX as if they were METAFONT fonts.

(LA)TeX supports handling the PostScript code as a literal in the `\spacial` command in different ways, depending on the scope of the code in the generated PostScript document. The most important thing is that the PostScript variables, in these literals, can be evaluated based on TeX variables whose values are determined at a given time and in a given context. This PostScript code can in particular be a dynamic (parameterized) PostScript Type 3 font.

The PostScript Type 3 font is specified in the file dynMath.tps. It is a font which respects the Type 3 specification but it is included in the macro primitive `\special` and having a global PostScript scope:

```
% Content of ‘‘dynMath.tps’’
\special{!
⟨PostScript Type 3 specification of dynmath font⟩
}
```

This is an interaction between LaTeX and PostScript in which the Type 3 font is inserted and seen throughout the document generated by LaTeX via the `dvips` driver.

3.2 Symbols in table and encoding

Any font (PostScript in particular) defines a table of its character layout: graphics and code. The Type 3 font in `dynMath.tps` is called `dynMath`. We used `cmex10` (see Table 1) to build the layout of `dynMath` symbols. The `dynMath` layout is shown in Table 2. Because `dynMath` is dynamic, the symbol appears only once in the table. However, the symbol is parameterized to meet the required dimensions in a given context.

The symbols coded from 70 to 97 in the `cmex10` layout table come in two graphic versions (one for the mathematical mode `\scriptstyle` and the other for `\displaystyle`). We think that these signs, of TeX math class one (large operators), will remain in these two size cases (obviously referenced by their relative (LA)TeX commands). They will not be supported in `dynMath` except for the integral signs $\oint$ and $\int$. This is because an integral sign with a height greater than or equal to the mathematical quantity to be integrated looks better than the opposite case.

In [5], four fonts are mainly identified by the values `\textfont0`, `\textfont1`, `\textfont2` and `\textfont3` as the METAFONT fonts `cmr10.mf` (family 0), `cmmi10.mf` (family 1), `cmsy10.mf` (family 2) and `cmex10.mf` (family 3) respectively. We are interested in dynamic (extensible) symbols. In (LA)TeX, dynamism is managed by using different fonts depending on the context. To explain the concept, we will use the symbols “(”, “⟨”, “$\surd$” and “$\widehat{}$”. We consider the contents of the file `plain.tex` as reference.

- `\delcode‘\(="028300`: This means that the parenthesis “(” is a delimiter, of which the smallest variant is taken from family 0 at position 28x (40 in decimal) and the wide variant is in family 3 at position 00x.
- `\def\langle{\delimiter"426830A }`: This defines the symbol “⟨” as a delimiter of class 4 (opening delimiter), accessible via the `\langle` macro. The smallest variant is in family 2 at position 68x (104 decimal) and the largest is in family 3 at position 0Ax (10 decimal).
- `\def\sqrt{\radical"270370 }`: This defines the radical symbol “$\surd$” as a variable symbol whose smallest variant is in family 2 at position 70x (112 decimal) and the large variant is in family 3 at position 70x, accessed as the `\sqrt` macro.
- `\def\widehat{\mathaccent"0362 }`: This defines the wide hat symbol “$\widehat{}$” by means of the command `\widehat`, of class 0 (ordinary) and of which the smallest variant is in the family 3 at position 62x (98 decimal).

For the left parenthesis symbol, the smallest variant is encoded in the font `cmr10.mf` at position 40. The large variant with its various standalone instances is encoded in the font `cmex10.mf` at positions 0, 16, 18 and 32. The compound version is built from characters in the same font `cmex10.mf` at positions 48, 64 and 66 (repetitive character). The font `dynMath` is dynamic and so any symbol, such as the parenthesis, must appear only once in the layout table. The code is that of the first occurrence of the symbol in `cmex10.mf`, i.e., position 0 (see Table 2). To parameterize the parenthesis and thus support dynamism, we consider the encoding of the smallest variant, that in `cmr10.mf` which is relative to family 0.

There are other symbols whose parameterization is based on their appearances in the `cmr10.mf` font such as “)”, “[”, “]”, etc. Concerning the symbols “⟨” and “$\surd$” for example, they are in positions 10 and 112 respectively in `dynMath` (see Table 2) and their parametrizations are taken from the font `cmsy10.mf` relative to family 2. As for the sign “$\widehat{}$”, its position in `dynMath` is that in `cmex10.mf` and its parameterization base is of the smallest variant and taken from the same font, namely `cmex10.mf`.

Roughly speaking, it’s the encoding bases of the small symbol variants that are parameterized to support dynamism. Consider a symbol S. Its appearance in `dynMath` in Table 2, is of the form S_f^c with c designating the layout order number and f representing the family used as a basis for parameterization. Specifically, the opening bracket, the opening angle bracket and the wide hat are shown in Table 2 as $[\,{}_0^2$, $\langle\,{}_2^{10}$ and $\widehat{}\,{}_3^{98}$.

We transformed the fonts `cmr10.mf`, `cmsy10.mf` and `cmex10.mf` using METAPOST to PostScript code at a body size of 1000 units, serving as a basis for parameterization of dynamic mathematical symbol encoding, via the following commands:

```
mpost ’&mfplain \mode=localfont; \
mag=100.375; input cmr10.mf’
mpost ’&mfplain \mode=localfont; \
mag=100.375; input cmsy10.mf’
mpost ’&mfplain \mode=localfont; \
mag=100.375; input cmex10.mf’
```

Table 1: Math extension font layout showing `cmex10`

	0	1	2	3	4	5	6	7
′00*x*	0 (	1)	2 [	3]	4 ⌊	5 ⌋	6 ⌈	7 ⌉
′01*x*	8 {	9 }	10 ⟨	11 ⟩	12 |	13 ‖	14 /	15 \
′02*x*	16 (	17)	18 (	19)	20 [	21]	22 ⌊	23 ⌋
′03*x*	24 ⌈	25 ⌉	26 {	27 }	28 ⟨	29 ⟩	30 /	31 \
′04*x*	32 (	33)	34 [	35]	36 ⌊	37 ⌋	38 ⌈	39 ⌉
′05*x*	40 {	41 }	42 ⟨	43 ⟩	44 /	45 \	46 /	47 \
′06*x*	48 ⎛	49 ⎞	50 ⎡	51 ⎤	52 ⎣	53 ⎦	54 ⎢	55 ⎥
′07*x*	56 ⎧	57 ⎫	58 ⎩	59 ⎭	60 ⎨	61 ⎬	62 ⎪	63 ⏐
′10*x*	64 ⎝	65 ⎠	66 ⎜	67 ⎟	68 ⟨	69 ⟩	70 ⨆	71 ⨆
′11*x*	72 ∮	73 ∮	74 ⨀	75 ⨀	76 ⨁	77 ⨁	78 ⨂	79 ⨂
′12*x*	80 ∑	81 ∏	82 ∫	83 ⋃	84 ⋂	85 ⨄	86 ⋀	87 ⋁
′13*x*	88 ∑	89 ∏	90 ∫	91 ⋃	92 ⋂	93 ⨄	94 ⋀	95 ⋁
′14*x*	96 ∐	97 ∐	ˆ 98	ˆ 99	ˆ 100	˜ 101	˜ 102	˜ 103
′15*x*	104 [	105]	106 ⌊	107 ⌋	108 ⌈	109 ⌉	110 {	111 }
′16*x*	112 √	113 √	114 √	115 √	116 ⎷	117 │	118 ┌	119 ‖
′17*x*	120 ↑	121 ↓	122 ╭	123 ╮	124 ╰	125 ╯	126 ⇑	127 ⇓

There's a special case (as there may be more to come) for the opening and closing brace symbols. These are not based on existing fonts for parameterizing, but have been newly designed to meet the metal-likeness concept.

3.3 Parameterizing

Dynamic symbols are parameterized in the font to meet extension requirements. Two categories of characters are identified, depending on whether the dynamic parts are delimited by *straight lines* or *curved lines*. Two types of stretching are identified:

Table 2: `dynMath` font layout

	0	1	2	3	4	5	6	7
$'00x$	$($ 0 0	$)$ 1 0	$[$ 2 0	$]$ 3 0	$\lceil$ 4 2	$\rceil$ 5 2	$\lfloor$ 6 2	$\rfloor$ 7 2
$'01x$	$\{$ 8	$\}$ 9	$\langle$ 10 2	$\rangle$ 11 2	$\vert$ 12 2	$\Vert$ 13 2	$/$ 14 0	$\backslash$ 15 2
$'02x$	16	17	18	19	20	21	22	23
$'03x$	24	25	26	27	28	29	30	31
$'04x$	32	33	34	35	36	37	38	39
$'05x$	40	41	42	43	44	45	46	47
$'06x$	48	49	50	51	52	53	54	55
$'07x$	56	57	58	59	60	61	62	$\updownarrow$ 63 2
$'10x$	64	65	66	67	68	69	70	71
$'11x$	$\oint$ 72 3	73	74	75	76	77	78	79
$'12x$	80	81	$\int$ 82 3	83	84	85	86	87
$'13x$	88	89	90	91	92	93	94	95
$'14x$	96	97	$\frown$ 98 3	99	100	$\sim$ 101 3	102	103
$'15x$	104	105	106	107	108	109	110	111
$'16x$	$\surd$ 112 2	113	114	115	116	117	118	$\Updownarrow$ 119 2
$'17x$	$\uparrow$ 120 2	$\downarrow$ 121 2	122	123	124	125	$\Uparrow$ 126 2	$\Downarrow$ 127 2

1. Line-based extension: this type of extension is easy and straightforward to support. Examples include the bracket symbol "[", the up arrow symbol "↑", etc.

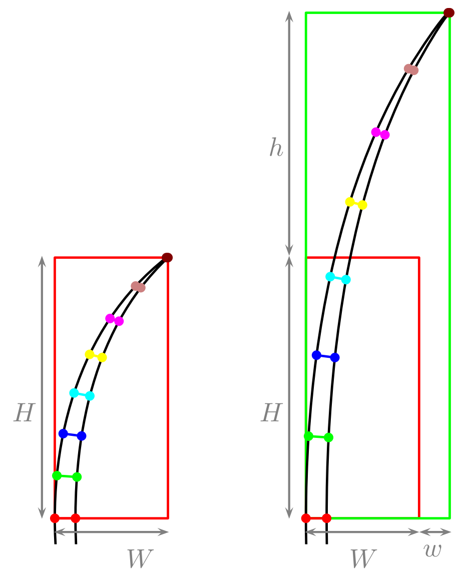
2. Curve-based extension: this extension concerns symbols whose dynamic parts have curved lines. Here, support for dynamism has necessitated the development of a mathematical stretching model (to be published) and an interpolation method that respects obliquity and convexity [3]. Examples include the parenthesis "(", the brace "{", etc.

A dynamic symbol is characterized by three parameters: *height* (including *depth*), *width* and *thickness*. The thickness is in some way linked to the characteristics of the writing instrument (pen) or drawing instrument (brush).

The stretching undergone by a dynamic symbol is partly supported by the `dynMath.sty` package and partly by the `dynMath.tps` font. Consider the dynamic symbol S. Let H_S, W_S and E_S be its *height*, *width* and *thickness* respectively. If the symbol is to be stretched by the amount h vertically and w horizontally, then the features in the stretched state will be $H_S + h$, $W_S + w$ and E_S as its *height*, *width* and *thickness* respectively. Thickness is not affected by the extension. It should be noted that the stretching supported by the font is not linear. We'll call it semi-optical because the thickness remains unchanged. Globally speaking, the thickness also changes, but this is the work of LaTeX and the PostScript interpreter.

The concept is clarified in Figure 1. This is the case of the opening parenthesis but we have considered just the upper half, to show how the font takes care of the stretching on its side. Note that the example is computed at 10 font size in TeX points but scaled linearly 20 times for greater clarity. It is as if the parenthesis at size 200 in TeX points undergoes stretching of the amounts w horizontally and h vertically.

The thickness was not affected by the stretching. To highlight this, we have considered landmarks with different colors. The upper half of the parenthesis is delimited by two sequences of curves, one on the left and the other on the right. Each sequence is made up of 7 cubic Bézier curves (how we get these sequences is a separate work from the current one). The points shown are the boundary control points of the Bézier curves. Points of the same order in both sequences are of the same color and linked by a segment also of this color. Our mathematical stretching model preserves the same convexity sense, obliquity and thickness. This is expressed by the fact that segments of the same color are of the same length and direction (in the vector sense) in Figure 1a and Figure 1b.



(a) Top left part of left parenthesis without stretching

(b) Top left part of left parenthesis with stretching of h vertically and w horizontally

Figure 1: Example of stretching in height h and width w while keeping the same thickness

4 `dynMath`: the style package

4.1 Useful macros and conventions

The `dynMath.sty` style package defines all the variables useful for internal operations, as well as others used as an interface for interaction with the PostScript Type 3 font `dynMath`. It also defines macros for managing mathematical formulas based on extensible symbols. We have followed a particular way of naming the macros relating to the dynamic symbols in LaTeX. In LaTeX, without doubt, the most interesting commands, in term of dynamism, are the primitive `\left` and its counterpart `\right`. The package `dynMath` defines a macro which essentially does the same job as `\left` but operates with the dynamic symbols defined in the PostScript Type 3 font. The general syntax of this macro is:

`\meLeft`⟨*delim*$_1$⟩ ⟨*formula*⟩ `\meRight`⟨*delim*$_2$⟩

We referred to the normal LaTeX commands when naming the `dynMath` ones in order to make it easier to use for users accustomed to using (LA)TeX. The same names are used, beginning with a capital letter and preceded by “`me`” meaning “metal” . Another example is `\overbrace`, to which corresponds `\meOverBrace` in `dynMath`.

5 Determining extension parameters

The most characteristic stretching parameters are the amount of vertical stretching h, the amount of horizontal stretching w and the size of the font PostScript f_s in which we will typeset the symbol to be stretched. In the case of the `\meLeft` macro, that is, in the case of the delimiters, these three parameters are functions of the mathematical height of a formula, which we will always call h_m. First, we give the idea of calculating h_m. Figure 2 and Figure 3 explain the approach. This concerns the case of two abstract mathematical formulas (just a rectangle with a height, depth and width) one of which is high and the other is deep. A description of the parameters in the figures are as follows:

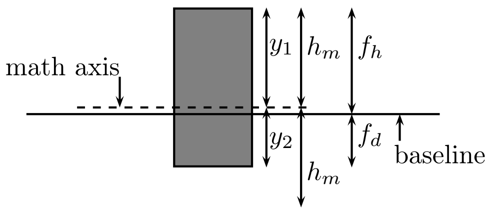


Figure 2: Abstract high mathematical formula

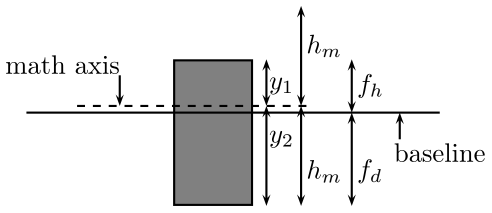


Figure 3: Abstract deep mathematical formula

- 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 $h_m = \text{Sup}\,(y_1, y_2)$.

An important point to note is that the handling of stretchable mathematical symbols differs from one category to another. For example, the parameter h_m, which makes sense in the case of delimiters, will not make sense when it comes to the radical (square

Table 3: Characteristics of left parenthesis in PostScript at a 10 unit body size

Parenthesis	Width	Close width	Height	Thickness	Left bearing	Right bearing	Math axis
Normal	3.8688232	2.3218745	5.0000024	0.5833423	0.9942598	0.5526889	2.5000014
`big`	4.5632816	2.6121009	5.9900241	0.6944529	1.5242052	0.4269755	
`Big`	5.9521679	3.7974070	8.9900563	0.7638956	1.8027775	0.3519834	
`bigg`	7.3410694	4.9213804	11.9900885	0.8333385	2.081079	0.33861	
`Bigg`	7.8966266	5.2128642	14.9901202	0.972224	2.3589455	0.3248169	
Compound	8.7299554	5.5043343	17.9801222	1.1111097	2.9145983	0.3110228	

root) symbol. For the radical, the amount of vertical stretching, for example, depends on the overall height including depth, i.e., $f_h + f_d$.

We only consider the case of the macro `meLeft`, since the aim is to illuminate the interaction between (LA)TEX and PostScript Type 3. The value of parameter h_m is half the overall height of the extensible delimiter. To clarify the idea, we take one of the previous figures, Figure 3 for this example, and display on it the left parenthesis useful for delimiting the abstract formula. The result is in Figure 4. The parenthesis delimiter is positioned correctly vertically. However, it has been shifted a little horizontally to the left to give the figure more legibility.

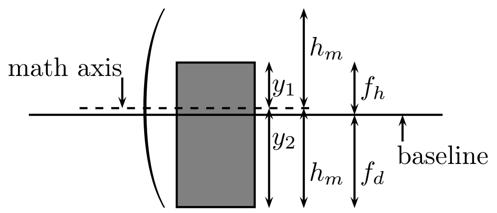


Figure 4: Abstract deep mathematical formula with left parenthesis delimiter

The opening (and closing) parenthesis in TEX comes in five standalone versions, as shown in Figure 5. One or the other is used to delimit a formula, depending on the situation of its mathematical height in comparison with those of the parentheses. When the mathematical height of a formula exceeds that of the standalone parentheses, a three-character compound parenthesis is used. This is made up of the three characters ⎛, ⎝ and ⎜, vertically superposed, with the third repeated between the two first as many times as necessary.

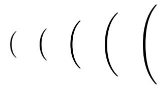


Figure 5: (LA)TEX standalone left parentheses

Let's adopt a numbered designation for the parentheses, P_0, …, P_5, in the order given in Figure 5. The last, P_5, is the smallest compound parenthesis, i.e., the compound when the number of occurrences of the repeated character is zero. The parenthesis P_0 represents the smallest variant in standalone parentheses (as we saw before). It is none other than parenthesis number 40 in `cmr10.mf`. It's this parenthesis that we've set in the PostScript Type 3 font `dynMath` to support dynamic parenthesis. Its encoding in PostScript is developed as a function of the two variables w and h (among others) representing horizontal and vertical stretching respectively.

Table 3 shows the most important characteristics of the six parentheses used in (LA)TEX. One notable parameter of the state of a parenthesis is the thickness e_m. How is this value calculated? In the case of a delimiter to be stretched relative to `\meLeft`, it's the stretching in the vertical direction that attracts attention. This shows that h_m is a key parameter in handling the dynamism of delimiters. For this purpose, thickness is defined as a function of the mathematical height h_m. Let $(h_{m,i})_{i=0}^{5}$ denote the sequence of mathematical heights of the parentheses $P_0, \dots, P_5$. Similarly, $(e_{m,i})_{i=0}^{5}$ is the sequence of the thicknesses of $P_0, \dots, P_5$. The 10pt size is taken as a basis for handling the stretching of the parenthesis symbol. We have the following cases and constraints:

- If $h_m = h_{m,i}$, $i = 0, \dots, 5$ then $e_m = e_{m,i}$.
- If $h_m \in [0, h_{m,0}]$, then e_m is linearly increasing between 0 and $e_{m,0}$.
- If $h_m \in [h_{m,i}, h_{m,i+1}]$, $i = 0, \dots, 4$ then e_m is increasing affinely between $e_{m,i}$ and $e_{m,i+1}$.
- If $h_m \in [h_{m,5}, h_{\max}]$, then e_m is increasing affinely between $e_{m,5}$ and $e_{\max}$.

The value of the maximum mathematical height taken is $h_{\max} = 1685$pt. This value represents approximately half the height of an A0 page. As for the thickness corresponding to $h_{\max}$, determined by experiments based on a certain formulation, it is $e_{\max} = 6.292214230$pt.

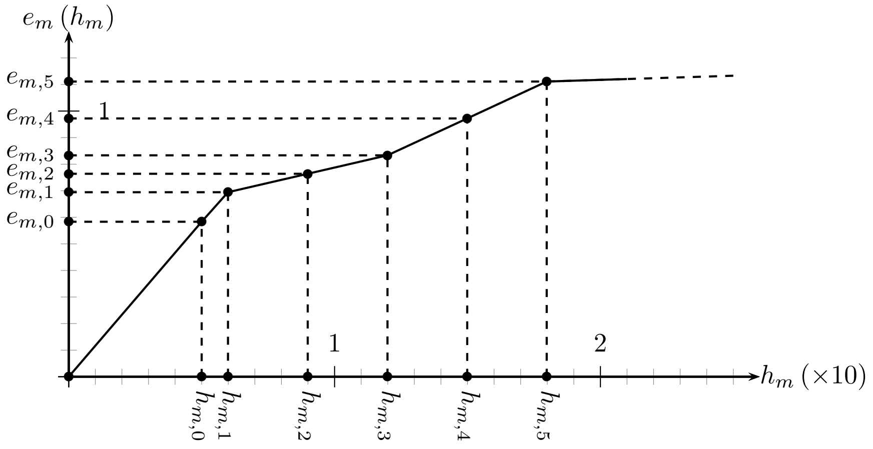


Figure 6: Thickness e_m as a function of h_m

A summary of the cases is shown in Figure 6. Remember that it's the parenthesis P_0, but with a 1000 unit body size, that is implemented and parameterized in the PostScript Type 3 font. What counts first when typesetting a mathematical symbol, such as the opening parenthesis, is the size f_s of the font. Assuming that for a mathematical height h_m, the thickness is e_m, knowing the thickness e_{1000} of the parenthesis at size 1000 in the PostScript font `dynMath`, then we can determine the size value f_s of the font corresponding to this thickness e_m, i.e., $f_s = {}^{(e_m \times 1000)}/_{e_{1000}}$.

Let a font size f_s correspond to a thickness e_m which we calculated as a function of h_m. Let us denote by h_{f_s} the height of the parenthesis in the font PostScript Type 3 `dynMath` relative to f_s, without any extension ($w = 0$, and $h = 0$). So we have:

1. If $h_m \le h_{m,0}$ then $h_{f_s} = h_m$
2. If $h_m > h_{m,0}$ then $h_{f_s} < h_m$.

We assume that h represents the amount of vertical stretching the parenthesis in `dynMath` must undergo to delimit the mathematical formula. For Item 1, the parenthesis obtained has the necessary height to cover the formula. There's no need to stretch this parenthesis, so $h = 0$. On the other hand, in Item 2, we need a vertical extension $h = h_m - h_{f_s}$ for the parenthesis to have the height needed to cover the formula. The horizontal stretching amount w needed will be explained later.

Just like the thickness e_m, other functions are useful and defined according to h_m: the width w_m, the strict or close width (width of the symbol without the left and right bearings) cw_m, the left bearing lb_m and right bearing rb_m.

The function cw_m is important for calculating the amount of horizontal stretching w. For this, we give some detail on its definition. The sequence $(cw_{m,i})_{i=0}^{5}$ consists of the close widths of the parentheses P_0 to P_5. We have:

- If $h_m = h_i$, $i = 0, \ldots, 5$ then $cw_m = cw_{m,i}$.
- If $h_m \in [0, h_{m,0}]$, then the function is of no interest (see further).
- $h_m \in [h_{m,i}, h_{m,i+1}]$, $i = 0, \ldots, 4$ then cw_m is increasing affinely between $cw_{m,i}$ and $cw_{m,i+1}$.
- If $h_m \in [h_{m,5}, h_{\max}]$, then cw_m is of no interest.

If we reconsider the font size f_s and denote by cw_{f_s} the close width of the parenthesis in the Type 3 font `dynMath` at size f_s, we get the following result: $cw_{f_s} < cw_m$. The horizontal stretching variable w takes on the following values:

1. If $h_m \le h_{m,0}$ then $w = 0$ (in this case $h = 0$, see Item 1 above).
2. If $h_{m,0} < h_m \le h_{m,5}$ then $cw_{f_s} < cw_m$ and $w = cw_m - cw_{f_s}$.
3. If $h_m > h_{m,5}$ then $w = \frac{h}{8}$. This is a relationship obtained by experimentation. It differs from one symbol to another. For the brace, for example, it's $w = \frac{h}{16}$.

For further clarification, two illustrations of the last two cases of the above enumeration are in Figures 7 and 8. These figures present information other than that relating to the stretching, vertical h and horizontal w. The meanings of the various parts were given in Figure 2 and Figure 3.

Processing of the left and right bearings is required to correctly position the dynamic symbols around the mathematical formula to be delimited. We need to be aware that the mathematical axis of the symbol written in the `dynMath` font is different from that of the mathematical formula, and so an alignment is necessary. We won't go into the details of these functions here, so as not to overload the article. In a future project, we'll write a book on the detailed implementation of `dynMath`.

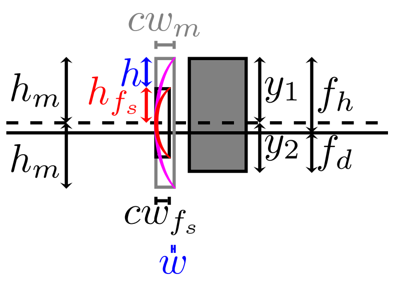


Figure 7: Stretching details, $H_4 < h_m < H_5$

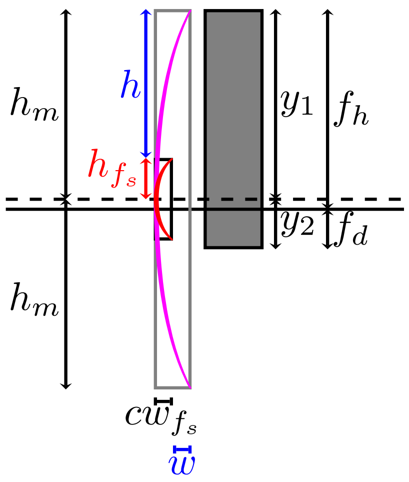


Figure 8: Stretching details, $h_m > H_5$

5.1 Dynamism management steps

In this section, the important steps in dynamism management are presented. It should be noted that each macro relating to the extension phenomenon is responsible for managing the relative extension parameters. The need may differ from one macro to another. Consideration of one of them highlights the general concept. The macro used as an example is `\meLeft`. One of the steps in the extension process is interaction with the Type 3 font. We are not going to talk about the `\meLeft` macro in programming terms, but only in an algorithmic sense and in a language as natural and abstract as possible. The definition of this macro is:

`\def\meLeft#1#2\meRight#3{`⟨ *macro definition*⟩`}`

Where:
`#1`: left delimiter,
`#2`: formula to be delimited,
`#3`: right delimiter.

Let's assume that:
`ldel`: represents `#1`,
`formula`: represents `#2`,
`rdel`: represents `#3`.

Before presenting the steps of the `\meLeft` macro, the meanings of some keywords used are given in Table 4.

Table 4: Useful tokens and meaning for dynamism steps presentation

Keyword	Meaning
`ldel`	`left delimiter`
`rdel`	`right delimiter`
`mAxis`	`mathematical Axis`
`fbox`	`formula box`
`fh`	`formula height`
`fd`	`formula depth`
`fw`	`formula width`
`hm`	`height mathematical`
`lth`	`left thickness`
`fs`	`font size`
`symWidth`	`symbol Width`
`fdelb`	`formula delimiter box`

The main steps of `\meLeft` are:

1. Determine the current math style: `style`
2. In `style`:
   - Determine the height of the mathematical axis: `mAxis`.
   - Put `formula` in `fbox`.
3. Determine the dimensions of `fbox`:
   - Height: `fh`
   - Depth: `fd`
   - Width: `fw`
4. Determine the mathematical height `hm`: `hm` $=$ sup (`fh` $-$ `mAxis`, `fd` $+$ `mAxis`)
5. Based on `hm`, determine the thickness of the left dynamic symbol `ldel`: `lth`.
6. Based on `lth`, determine the size `fs` of the PostScript font `dynMath` to write the delimiter `ldel`.
7. In terms of `fs` and `hm` determine:
   - The vertical stretching amount `h`.
   - The horizontal stretching amount `w`.
   - The delimiter width `symWidth`.
8. Process the box `fdelb` which will contain the extensible PostScript delimiter:
   - Write in `fdelb` the special: `\special{"` ⟨*leftSpecial*⟩`}`.
   - In ⟨*leftSpecial*⟩:
     - Align the mathematical axis of the symbol `ldel` according to the font `dynMath` at size `fs` with the mathematical axis `mAxis` of `formula`.
     - Write `ldel` with respect to the font `dynMath` at size `fs` from the coordinates $(0, 0)$.

9. Set the dimensions of `fdelb`:
    - Width at `symWidth`.
    - Height at (`hm` + `mAxis`).
    - Depth at (`hm` − `mAxis`).
10. Adjust the position of `fdelb` by kerning in order to adjust the left bearing of `ldel`.
11. Insert the contents of the `fdelb`.
12. Adjust the right bearing of `ldel` by kerning.
13. Insert `formula`.
14. Repeat steps 5 to 12 for the `rdel` delimiter.

6 Conclusions

We have given an idea on the principles of interaction between (LA)TEX and a PostScript Type 3 font. This is the basis for the support of dynamic mathematical symbols. The idea is presented in special cases and not completely detailed. In the near future, we will publish a book detailing the basics and all the implementation cases of `dynMath`.

References

[1] Adobe Systems Incorporated. *PostScript Language Reference Manual*. Adobe Systems Incorporated, Addison-Wesley Publishing Company, Reading, Massachusetts, 1999. `https://adobe.com/jp/print/postscript/pdfs/PLRM.pdf`.

[2] A. Bayar. Towards an operational (LA)TEX package supporting optical scaling of dynamic mathematical symbols. *TUGboat* 37(2):171–179, 2016. `https://tug.org/TUGboat/tb37-2/tb116bayar.pdf`

[3] A. Bayar. C^1 interpolation of sequences of points preserving convexity and obliquity based on oblique convex two-dimensional cubic bézier splines. In *2024 IEEE International Conference on Signal, Image, Video and Communications (ISIVC 2024)*, pp. 1–6, Marrakech, Morocco, May 2024.

[4] A. Bayar. `dynMath`: A PostScript Type 3-based LATEX package to support extensible mathematical symbols. *TUGboat* 45(1):18–24, 2024. `https://tug.org/TUGboat/tb45-1/tb139bayar-dynmath.pdf`

[5] D.E. Knuth. *The TEXbook*, vol. A of *Computers and Typesetting*. Addison-Wesley, Reading, Massachusetts, 1st ed., 1984.

[6] L. Lamport. *LATEX: A Document Preparation System*. Addison-Wesley, USA, 1994.

[7] LuaTEX development team. *LuaTEX Reference Manual*, Feb. 2024. `https://ctan.org/pkg/luatex`.

[8] T. Rokicki. *Dvips: A DVI-to-PostScript Translator*, Feb. 2024. `https://ctan.org/pkg/dvips`.

◇ Abdelouahad Bayar
Cadi Ayyad University — Higher School of Technology of Safi, PSSII Lab
Sidi Aissa Road, PB 89
Safi, 46000
Morocco
`a.bayar (at) uca dot ma`
ORCID 0000-0002-3496-505X