Aligning text in diagrams exported by Mathematica: A question about the PostScript infrastructure

Michael P. Barnett

Abstract

I produce many LaTeX documents that contain diagrams exported by Mathematica* graphics. Usually, these contain text. Often, this is misaligned horizontally. I think that getting correct alignment needs an understanding of PDF font encoding. This note describes the problem in hope of getting feedback.

1 Introduction

This note seeks advice from PostScript experts about certain details of font encoding. I need this information to align built-up text expressions that mix different fonts, in diagrams that are constructed by Mathematica graphics. I include these diagrams in LaTeX manuscripts on topics in the natural sciences, mathematics and the humanities. The text is aligned by the TMG (text in Mathematica) package that I wrote. Fig. 1 and nearly 30 similar diagrams are in a recent paper on nuclear magnetic resonance (NMR) that I wrote with István Pelczer [1]. The construction

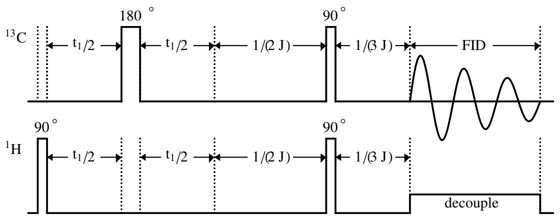

Figure 1: An NMR pulse sequence diagram.

of these diagrams prompted the work on TMG. The package contains an `encode` function that tries to position the contents of separate Mathematica `Text` commands precisely. This is a standard need when a set of related diagrams consists of varied selections of modules that contain text. The definitive description of the `Text` command in *The Mathematica Book* [2] states:

“**Text**[*expr, coords, offset*] specifies an offset for the block of text relative to the coordinates given.”

The description goes on to mention sample offsets that include “$\{-1, 0\}$ left-hand end at $\{x, y\}$” and “$\{0, -1\}$ centered above $\{x, y\}$”. The obvious extension is that $\{-1, -1\}$ puts the lower left corner of the text at $\{x, y\}$. The description in [2] refers to the “bounding rectangle that surrounds the text”.

The idea of bounding rectangles that surround text has been inherent in the use of moveable type for millennia [3] and, more recently, in phototypesetting [4]. It is associated with the idea of a baseline, defined as “the line upon which most letters ‘sit’ and below which descenders extend” [5]. Fig. 2 shows a sequence of words and isolated characters in serif, sans-serif and Greek fonts, that were typeset by elementary LaTeX coding. The rectangles that surround the characters and the baseline were drawn by `\rule` commands. Typesetting software has customarily

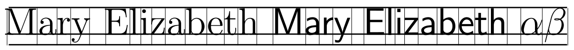

Figure 2: Bounding boxes — consistent baselines.

treated the vertical coordinate of a piece of text as the position of its baseline, since the inception of the field in the late 1950s [4]. Digital fonts were developed that treated each character as if it were contained in a rectangle, that had the point size as its height, with baselines positioned for consistency between characters in the same font and in different fonts. This paralleled the design of metal type slugs.

Fig. 3 shows some bad alignment produced by Mathematica `Text` commands. These all contain the offsets `{-1,-1}`, and the same y value is used in the `Text` and `Line` commands that produced each row. `Line[`$\{\{x_1, y_1\}, \{x_2, y_2\}\}$`]` draws a line from (x_1, y_1) to (x_2, y_2).

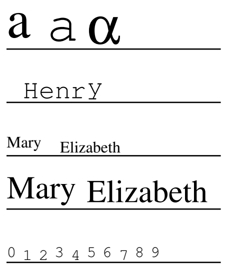

Figure 3: Examples of unexpected misalignment.

```
{Text[Style["a", FontFamily -> "Times-Roman",
  FontSize -> 20], {0,600}, {-1, -1}],
 Text[Style["a", FontFamily -> "Courier",
  FontSize -> 20], {15, 600}, {-1, -1}],
 Text[Style["a", FontFamily -> "Symbol",
  FontSize -> 20], {30, 600}, {-1, -1}],
 Line[{{0, 600}, {80, 600}}],
(*  *)
 Text[Style["H", FontFamily -> "Courier",
  FontSize -> 10], {6, 580}, {-1, -1}],
 Text[Style["e", FontFamily -> "Courier",
```

* MATHEMATICA is a registered trademark of Wolfram Research Inc.

```
  FontSize -> 10], {12, 580}, {-1, -1}],
...
 Text[Style["y", FontFamily -> "Courier",
  FontSize -> 10], {30, 580}, {-1, -1}],
 Line[{{0, 580}, {80, 580}}],
(*  *)
 Text[Style["Mary",
  FontFamily -> "Times-Roman", FontSize -> 6],
  {0, 560}, {-1, -1}],
 Text[Style["Elizabeth",
  FontFamily -> "Times-Roman",
  FontSize -> 6], {20, 560}, {-1, -1}],
 Line[{{0, 560}, {80, 560}}],
(*  *)
 Text[Style["Mary",
  FontFamily -> "Times-Roman",
  FontSize -> 12], {0, 540}, {-1, -1}],
 Text[Style["Elizabeth",
  FontFamily -> "Times-Roman",
  FontSize -> 12], {30, 540}, {-1, -1}],
 Line[{{0, 540}, {80, 540}}],
(*  *)
 Text[Style["0", FontFamily -> "Courier",
  FontSize -> 6], {0, 520}, {-1, -1}],
 Text[Style["1", FontFamily -> "Courier",
  FontSize -> 6], {6, 520}, {-1, -1}],
...
 Text[Style["9", FontFamily -> "Courier",
  FontSize -> 6], {54, 520}, {-1, -1}],
 Line[{{0, 520}, {80, 520}}]}];
```

An earlier version of these commands is shorter but needs more explanation. There are several reasons for the alignment effects in Fig. 3.

1. In the 1st row, the letter "a" in Times-Roman and Courier fonts and the α do not line up because the different fonts are coded with different baselines in their respective bounding boxes.
2. In the 2nd row, the *bases* of the rectangles that fit tightly around the individual letters in "Henry" are aligned. This makes the "y" high relative to the other letters.
3. In the 3rd and 4th rows, the string "Mary" has consistent baselines. So does "Elizabeth". But the "y" in "Mary" pushes the entire string up, relative to "Elizabeth".
4. In the 5th row, I think that the digits do not line up because 0, 3, 5, 6, 8 and 9 were coded using one set of conventions, and 1, 2, 4 and 7 using a different set.

Fig. 4 shows a practical consequence of the alignment of personal names. I wrote a Mathematica script in the 1990s to display genealogies. Fig. 4 is a minimalistic display of relationships that dominated British society for half a century. The misalignment would be unacceptable in a scholarly journal that dealt with the substantive issues that this presents.

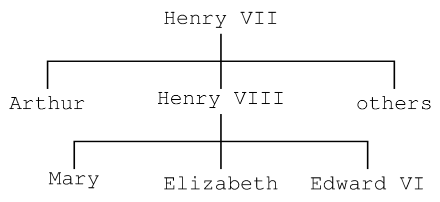

Figure 4: The simplified Tudor succession.

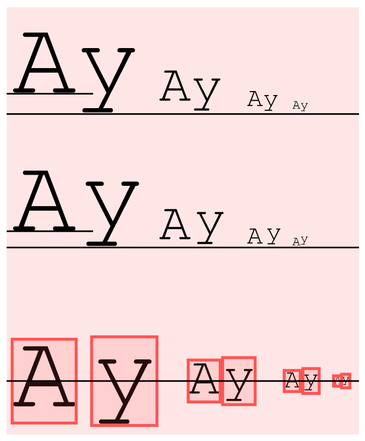

Figure 5: Forced demonstration of bounding box.

Some forms of undesirable alignment become more pronounced as font size decreases. This may be related to an apparent drift in the snugness of the bounding rectangle. Although I have not found relevant data directly, some simple syntactic errors make the system display the rectangles that it seems to use. This is done in the 3rd row of Fig. 5 by using `null` as the y offset. The system treats it as 0. In the 1st row, the string "Ay" is set successively in 40, 20, 10 and 5 point Courier type. The y coordinate in the `Text` expressions is 550, and a `Line` expression draws a line with $y = 550$ across the page. The serif of the "y" touches this line in 40 point type, but not in the smaller sizes. A line drawn with $y = 557.6$ shows the elevation of the serifs of the "A" relative to those of the "y" in 40 point type.

In the 2nd row, the two letters "A" and "y" are set by separate `Text` statements, with $y = 500$. The serif of the 40 point "y" touches a line with this coordinate, but the relative elevation of the "A" has dropped to 6.1 points. As the size decreases, the "y" continues to move up relative to the "A".

In the 3rd row the rectangles surrounding the "y" have moved down slightly, with decreasing point size, relative to the serif. Fig. 6 shows the 5 point example, magnified 8-fold by the `scale` parameter in the LaTeX

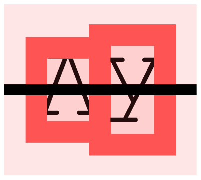

Figure 6: 8-fold magnification of 5 point example.

`\includegraphics` command. The position of the horizontal line at $y = 450$ emphasizes the change.

2 The TMG encode function

I try to achieve the horizontal alignment of a body of text that is displayed on a single line by putting the i-th character, denoted here by c_i, into a separate expression of the form

`Text[Style[`c_i`,FontFamily->`f_i`, FontSize->`s_i`],`
`{`$x_0 + h_i,\ \bar{y}$`},{-1,`$\sigma(f_i, s_i, c_i)$`}]`

where

1. $h_i = \sum_{j=1}^{i-1} \frac{s_j}{10} w(f_j, c_j)$,
2. c_i, f_i and s_i are the i-th character and the font style and font size in which it is set,
3. $w(f_i, c_i)$ is the width of c_i in 10 point font f_i (giving $w($`"Courier"`$, c)$ the value 6 for the entire character set),
4. $\sigma(f_i, s_i, c_i)$ is the offset that puts the baseline of the character, in the specified size and style, onto the line $y = \bar{y}$, that is specified in the coordinates part of the `Text` statement,
5. x_0 is the starting x coordinate of the text.

I developed methods to find widths and offsets by trial and error. Using these, I found the widths for the Courier, Times-Roman and Symbol fonts with ease and accuracy. I found the offsets for the Courier and Symbol fonts for point sizes 4 to 10, and Times-Roman for size 12, with considerable difficulty and tedium and some uncertainty.

I would like advice on finding the offsets algorithmically from the font tables.

The information may be in the chapter on fonts in the *PostScript Language Reference* manual [6]. The learning curve for this seems non-trivial, and I do not want to climb it unnecessarily.

The TMG expression

`encodeString[`*font*, *size*, x, y, *string*`]`

sets *string* in the specified font face and font size, starting with the left edge of the bounding box of the 1st character at x, and the baseline at y. In the more general expression

`encodeSequence[`$item_1, item_2, \ldots$`]`

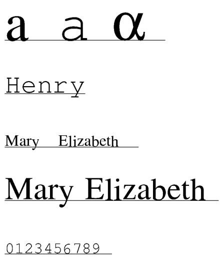

Figure 7: Output of encode expressions.

each item is either

1. a character string, *e.g.* “delay”, that is set in uniform font and size on a common baseline;
2. a character sequence with no quote marks, *e.g.* delay, that the system envelops in quote marks and treats as just described (this 2nd kind of item actually is restricted to objects that in Mathematica syntax are symbols);
3. one of the following commands
   (a) `ps[`n`]`: changes font to size n without altering the baseline,
   (b) `tf[`f`]`: changes the font to style f,
   (c) `tf[`f, n`]`: changes the font to style f and size n,
   (d) `sub[`s`]`: sets the string s in the decoration size, sunk to subscript level,
   (e) `sup[`s`]`: sets s in decoration size, raised to superscript level,
   (f) `subSup[`s_1, s_2`]`: sets s_1 and s_2 as subscript and superscript, left aligned,
   (g) `lSubSup[`s_1, s_2`]`: sets s_1 and s_2 as right aligned subscript and superscript,
   (h) `tab[`$\bar{x}$`]`: changes the x coordinate for the next displayed object to $\bar{x}$,
   (i) `vtab[`$\bar{y}$`]`: changes the y coordinate for the next displayed object to $\bar{y}$,
   (j) `hs[`$\bar{n}$`]`: increases x by n,
   (k) `vs[`$\bar{n}$`]`: increases y by n.

I hope to extend this set of commands to provide algorithmic formatting capabilities.

The file `alignedByEncode.pdf` that produced Fig. 7 for comparison with Figs. 3 and 4 was written by the following statement, that contains `encode` and `encodeString` expressions.

```
export[alignedByEncode =
{AbsoluteThickness[.1],
 encode[ps[20], vtab[620], tab[20],
  tf["Times-Roman"], "a", tab[40],
  tf["Courier"], "a", tab[60],
  tf["Symbol"], "a"],
 Line[{{20, 620}, {80, 620}}],
 encodeString["Courier", 10, 20, 600,
  "Henry"],
 Line[{{20, 600}, {50, 600}}],
 encodeString["Times-Roman", 6, 20, 580,
  "Mary"],
 encodeString["Times-Roman", 6, 40, 580,
  "Elizabeth"],
 Line[{{20, 580}, {70, 580}}],
 encodeString["Times-Roman", 12, 20, 560,
  "Mary"],
 encodeString["Times-Roman", 12, 50, 560,
  "Elizabeth"],
 Line[{{20, 560}, {100, 560}}],
 encode[tf["Courier"], ps[6], tab[20],
    vtab[540], "0", "1", "2", "3", "4", "5",
    "6", "7", "8", "9"],
 Line[{{20, 540}, {60, 540}}]}]
```

The file `refinedTudors.pdf` that produced Fig. 8 was written by the following statements.

```
tudorTreeEdges =
 {Line[{{200, 600}, {200, 590}}],
  Line[{{135, 590}, {265, 590}}],
  Line[{{135, 590}, {135, 580}}],
  Line[{{200, 590}, {200, 580}}],
  Line[{{265, 590}, {265, 580}}],
  Line[{{200, 570}, {200, 560}}],
  Line[{{145, 560}, {255, 560}}],
  Line[{{145, 560}, {145, 550}}],
  Line[{{200, 560}, {200, 550}}],
  Line[{{255, 560}, {255, 550}}]}

encodedTudorNames =
{encodeString[
  "Courier", 8, 178.4, 605, "Henry VII"],
 encodeString[
  "Courier", 8, 120.6, 575, "Arthur"],
 encodeString[
  "Courier", 8, 186.0, 575, "Henry VIII"],
 encodeString[
  "Courier", 8, 250.6, 575, "others"],
 encodeString[
  "Courier", 8, 135.4, 545, "Mary"],
 encodeString[
  "Courier", 8, 178.4, 545, "Elizabeth"],
 encodeString[
  "Courier", 8, 233.4, 545, "Edward VI"]}

export[refinedTudors =
  {tudorTreeEdges, encodedTudorNames}]
```

The alignment is imperfect but I believe it can be improved by fine tuning the offsets. I think that each letter has a range of offsets that are acceptable in one context, and a different range in another context, with very narrow overlap. I have been using just one or two contexts to determine the offsets for each letter, and picking an offset within the range of acceptability in a somewhat arbitrary manner.

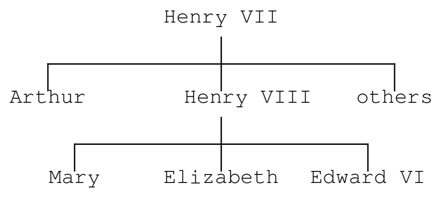

Figure 8: More output of encode expressions.

The present technique supports the alignment of short expressions with each other, as needed in the pulse sequence diagram of Fig. 1. That diagram, and the other diagrams in [1], were produced by *ad hoc* coding before I started TMG. Using TMG, I will be able to extend the options for including explanatory text in the diagrams, even in its present crude form. An algorithmic basis for TMG would enable many other applications of Mathematica graphics in the kernel mode.

Acknowledgements

I thank Barbara Beeton, Karl Berry and Andrew Roberts for advice on coding and on sources of information.

Supplementary material

Accompanying this article on the *TUGboat* web site is a collection of additional material:

1. the TMG software described in this note,
2. an account of how to measure widths and offsets,
3. software that I used to explore Mathematica fonts, with a detailed explanation.

References

[1] M. P. Barnett and I. Pelczer, Pulse sequence editing by symbolic calculation, J. Magn. Reson. 204 (2010) 189–195.

[2] S. Wolfram, *The Mathematica Book*, 2nd ed. Addison-Wesley, New York, 1991, or later editions.

[3] Movable type. `http://en.wikipedia.org/wiki/Movable_type`.

[4] M. P. Barnett, *Computer Typesetting, Experiments and Prospects*, MIT Press. 1965.

[5] Baseline (typography). `http://en.wikipedia.org/wiki/Baseline_(typography)`.

[6] PostScript Language Reference, Adobe Systems Incorporated. `http://www.adobe.com/products/postscript/pdfs/PLRM.pdf`.

◇ Michael P. Barnett
Meadow Lakes
Hightstown, NJ 08520, USA
`michaelb (at) princeton dot edu`
`http://www.princeton.edu/~michaelb/nmr/`