Axis alignment in XY-pic diagrams

Alexander R. Perlis

Abstract

By default, XY-pic aligns diagrams according to the *center* of each object. For many types of diagrams, such *center alignment* is the preferred choice; however, *axis alignment* is sometimes better. For example, compare $A \twoheadrightarrow A^2$ (center-aligned) with $A \twoheadrightarrow A^2$ (axis-aligned); the arrow stayed in the same place, but the A moved up a little, and the A^2 moved up a lot. Note that the simple TEX code `$A \to A^2$` uses axis alignment: $A \to A^2$.

This article studies attempts to instruct XY-pic to use axis alignment and presents a concise solution to the problem. Enhancements to XY-pic are proposed.

1 Preliminaries

The version of XY-pic used here is 3.7 (16Feb1999). The *XY-pic User's Guide* will be cited as [XY GUIDE], and the *XY-pic Reference Manual* as [XY MANUAL]. These documents are part of the XY-pic distribution available on CTAN.

This article's abstract already defined *center alignment* and exhibited the subtle differences between that and *axis alignment*, but the latter was left undefined. For now, it means "the alignment used by the simplest of TEX code". Matters will make more sense after section 5, where the alignment practices of TEX and XY-pic are explained in detail.

For most of this article, we will study attempts at axis alignment using the `\xymatrix` feature of XY-pic. (It is introduced nicely in [XY GUIDE].) The *one-line solution* to our alignment problem appears in section 7. Solutions for the `\xygraph` feature and for XY-pic kernel code appear there as well.

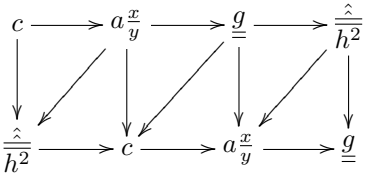
2 The problem

We are (happily) compelled to use XY-pic because it produces fantastic diagrams we cannot otherwise obtain, yet we recognize that many of our center-aligned diagrams ought to be axis-aligned. We wish to instruct XY-pic to use the preferred alignment.

To study the matter, we need a toy example that exhibits a wide gulf between the two types of alignment, yet fits in this article's columns.

```
\def\toyone{c} \def\toytwo{a\frac{x}{y}}
\def\toythree{\underline{\underline{g}}}
\def\toyfour{\hat{\hat{\overline
                      {\overline{h^2}}}}}
\def\toyexample{%
\toyone      \ar[d] \ar[r]
 & \toytwo   \ar[d] \ar[r] \ar[dl]
 & \toythree \ar[d] \ar[r] \ar[dl]
 & \toyfour  \ar[d] \ar[dl] \\
\toyfour     \ar[r]
 & \toyone   \ar[r]
 & \toytwo   \ar[r]
 & \toythree \\
}
```

The result of `\xymatrix{\toyexample}` is



What a monstrosity! To shirk responsibility, let's take the viewpoint that a famous mathematician has hired us to typeset this bewildering diagram as part of her new book. The notation is beyond our control.

A quick peek (go ahead!) at the end of section 7 shows what we're after. To get a sense of the difference, compare the top rows of the two diagrams without the arrows:

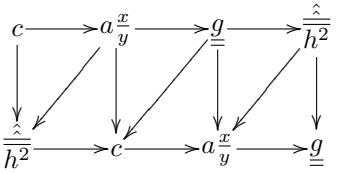
$$c\,a\tfrac{x}{y}\,\underline{\underline{g}}\,\hat{\hat{\overline{h^2}}} \quad \text{versus} \quad c\,a\tfrac{x}{y}\,\underline{\underline{g}}\,\hat{\hat{\overline{h^2}}}.$$

The reader interested merely in the solution to our problem should skip directly to section 7, or to section 5 for some background on that solution.

The material below in sections 3 and 4 has nothing to do with the ultimate solution, and is included mainly for the XY-pic enthusiast interested in why `\xymatrix@1` and variations on that theme do not solve our problem.

3 Using `\xymatrix@1`

The first idea is to try `\xymatrix@1` in place of `\xymatrix`. This gives



The top row is

$$c\,a\tfrac{x}{y}\,\underline{\underline{g}}\,\hat{\hat{\overline{h^2}}}.$$

That's neither center-aligned nor axis-aligned! To be fair, [XY GUIDE, §1.4, p.3] only encourages use of `\xymatrix@1` with one-line diagrams. Leaving our toy example aside for the rest of this section, let's

experiment a bit with one-line diagrams:[1]

```
$\xymatrix{A \ar[r] & A'}$
```
$A \to A'$

```
$\xymatrix@1{A \ar[r] & A'}$
```
$A \to A'$

Hoorah! The latter is axis-aligned! But if we replace A' with A^2, we get:

```
$\xymatrix{A \ar[r] & A^2}$
```
$A \to A^2$

```
$\xymatrix@1{A \ar[r] & A^2}$
```
$A \to A^2$

Unhoorah. This time the result is neither center-aligned nor axis-aligned: the placement of A^2 is too low. The mistake is easier to spot by enlarging and putting all the diagrams on a rule:

$$A \to A' \quad A \to A' \quad A \to A'$$

$$A \to A^2 \quad A \to A^2 \quad A \to A^2$$

Left: `\xymatrix`. Middle: `\xymatrix@1`. For comparison, `$A \to A'$` and `$A \to A^2$` are included on the right—being the simplest of TEX code, they are axis-aligned by definition!

In one case of a one-line diagram, `\xymatrix@1` succeeds, yet in another, it fails. Why did that happen? The difference between `\xymatrix` and `\xymatrix@1` is that the latter inserts a zero-width left parenthesis at the start of every entry, and doing so affects the vertical spacing.[2]

$$A \to A' \quad \big(A \to \big(A' \quad A \to A'$$

$$A \to A^2 \quad \big(A \to \big(A^2 \quad A \to A^2$$

An entry's center is determined by its bounding box, which in turn is determined by the parts of the entry that stick out the most. Both A and A' are dwarfed by the parenthesis, but A^2 is not: the superscript sticks out more than the parenthesis. Consequently, the center of $(A^2$ is slightly higher than that of $(A$ or $(A'$; thus, to align entries by their center, XY-pic must lower $(A^2$ slightly. (Why doesn't it instead raise everything else? We'll answer this question in section 5.)

In summary, `\xymatrix@1` gives an axis-aligned result only when the diagram's entries fit inside regular parentheses. Otherwise, the result likely will be neither center-aligned nor axis-aligned.

[1] The diagrams in `$...$` were made smaller by setting up `\xymatrix` with `@-1.25pc@M=1pt`. The sample code does not reflect this.

[2] There is another difference between `\xymatrix` and `\xymatrix@1`, not documented in [XY MANUAL]: `@1` implies `@M=1pt`. This explains why, in comparing the diagram at the start of section 3 to the one at the end of section 2, the arrows are closer to the entries.

4 Mimicking `\xymatrix@1`

Ah ha! With large entries, the zero-width left parenthesis inserted by `\xymatrix@1` is not tall enough, so let's use a taller one. Since

```
\xymatrix@1{\toyexample}
```

is equivalent[2] to

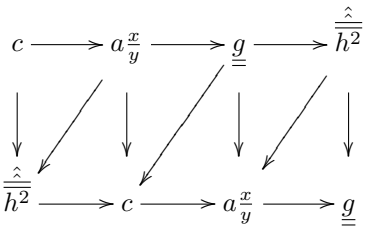
```
\everyentry={\vphantom(}
\xymatrix{\toyexample}
```

we might first try

```
\everyentry={\vphantom{\bigl(}}
\xymatrix{\toyexample}
```

but discover this isn't enough, and after running out of named sizes, we might try letting TEX make the precise calculation:

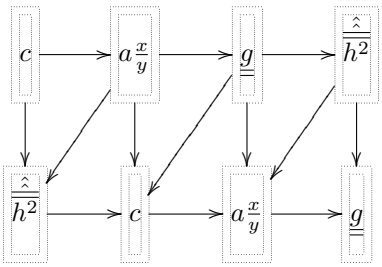
```
\everyentry={\vphantom{\left(\toyone\toytwo
                \toythree\toyfour\right.}}
\xymatrix{\toyexample}
```



Hey, this is axis-aligned! Unfortunately, many of the arrows now appear to be afraid of the entries. To understand what went wrong, take a look at the size of the delimiter we inserted around each entry:

```
\left(\toyone\toytwo
    \toythree\toyfour\right.
```
$\left(\begin{matrix} c & a\frac{x}{y} & \underline{g} & \hat{\hat{\bar{h^2}}} \end{matrix}\right.$

Each entry's vertical size is determined by the delimiter, and then, as usual, `\xymatrix` adds an additional margin:



Evidently, the promising approach of inserting zero-width material to affect vertical alignment, which is used by `\xymatrix@1`, is *fundamentally flawed*: important height information gets lost!

It's time to step back and study the alignment algorithms in TEX and XY-pic.

5 How TEX and XY-pic align objects

Earlier we asked: in going from $A \twoheadrightarrow A'$ to $A \twoheadrightarrow A'$, why does the arrow stay put and the A and A' move up, instead of, say, the A staying put, the arrow moving down, and the A' moving (slightly) up?

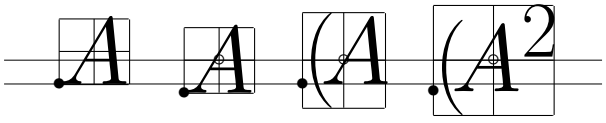
Inside math mode, TEX maintains two reference lines for alignment purposes: the *baseline* and the *axis*. These lines are not part of the characters being typeset; rather, they depend only on the current font and thus should be thought of as being part of the underlying canvas. As for the characters, each one has a bounding box and a reference point (obtained from the font's TFM file).

In horizontal mode and in math mode, TEX positions each character so that the character's reference point lands on the canvas's baseline. The axis comes into play when TEX builds a fraction: the numerator and denominator are positioned so that the bar of the fraction lands on the canvas's axis. Each delimiter, such as the left parenthesis, is designed to involve both lines: as with all characters, TEX positions the delimiter so that its reference point lands on the canvas's baseline; however, in so doing, due to the shape of the delimiter, the middle of the delimiter lands precisely on the canvas's axis. In other words, after placement, each delimiter has equal height and depth *when measured from the axis*, but not when measured from the baseline.

XY-pic, on the other hand, maintains its own reference point for each object, which starts out in the *center* of the object. When XY-pic hands a finished object to TEX for placement, it does so in such a way that the object's XY-pic reference point lands on the canvas's axis.[3]

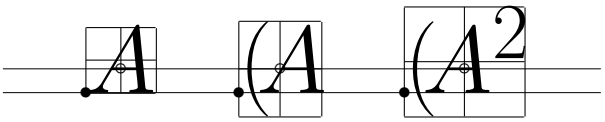
Let's summarize. When TEX is in charge, the TEX reference point lands on the baseline. When XY-pic is in charge, the XY-pic reference point lands on the axis.

Let's illustrate. On the left, we see how TEX positions an A: the TEX reference point • lands on the baseline.



Next we see how XY-pic positions A, $(A$, and $(A^2$: the XY-pic reference point ∘, which defaults to the object's center, lands on the axis. In the case of $(A$, the TEX reference point happens to land on the baseline, but that's merely a consequence of how delimiters are designed (discussed earlier in this section).

We wish to achieve the following:



The XY-pic reference point should be positioned away from the center in such a way that when it lands on the axis, the TEX reference point will land on the baseline. The crucial measurement is easy to spot: the vertical distance between the XY-pic reference point and the TEX reference point should be the same as the distance between the canvas's baseline and axis. TEX will cough up that distance if you feed it `\fontdimen22\textfont2`.

By positioning the XY-pic reference point appropriately, we achieve the desired alignment without changing the object's bounding box. This is the solution we've been seeking! The code is presented in section 7.

6 Aside: the term *axis-aligned*

By the discussion in the previous section, we conclude that *axis-aligned* means: objects are aligned with their TEX reference point on the baseline, while diagram arrows point to the axis (because that's where the XY-pic reference point is). But the following description does a better job justifying the term. Before being dropped on the canvas, each object is typeset in its own box and thus has its own axis. Getting the object's TEX reference point onto the canvas's baseline is equivalent to getting the object's axis onto the canvas's axis. Thus *axis-aligned* means: each object's axis lies on the canvas's axis, and each arrow points to that common axis. In short, everything is aligned by the axis!

7 The solution

We have seen that XY-pic positions an object so that the XY-pic reference point, which defaults to the object's center, lands on the canvas's axis. To alter the placement, we either move the object's center by changing its size, or move the XY-pic reference point away from the center. In section 4 we disposed of the idea of changing the object's size, because the original size is needed later for drawing arrows. At the end of section 5, we saw that moving the XY-pic reference point appropriately solves our problem.

All said and done, our solution is to put

```
\entrymodifiers={+!!<0pt,\fontdimen22\textfont2>}
```

prior to each `\xymatrix`, or simply once and for all in the document's preamble.

[3] As explained in [XY MANUAL, §2.1, p.6], that's true only when XY-pic was entered inside math mode. Otherwise there is no axis, and so the canvas's baseline is used. This affects *everything*. The net effect is to shift the entire diagram, arrows and all, by a fixed amount.

The `+` sets up a margin similar to the default margin used by `\xymatrix` (the difference is discussed briefly in section 9). The first `!` moves the Xy-pic reference point from the object's center down to the line containing TEX's reference point, and then `!<0pt,\fontdimen22\textfont2>` moves it up the appropriate distance so that, when it is dropped on the canvas's axis, the TEX reference point lands on the canvas's baseline.

Although [XY GUIDE] introduces Xy-pic in terms of the `\xymatrix` feature, there are other ways of using Xy-pic, notably `\xygraph` or even direct kernel code. With the `\xygraph` feature, axis alignment is obtained by putting

```
!~*{+!!<0pt,\fontdimen22\textfont2>}
```

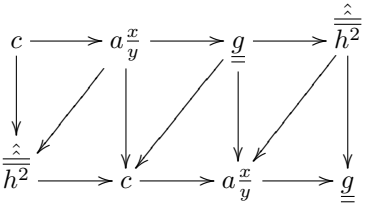
at the start of each graph. With Xy-pic kernel code, add the drop modifiers

```
!!<0pt,\fontdimen22\textfont2>
```

to each object that should be axis-aligned. But be careful: the effect is cumulative. Thus, if you drop a `\composite` of objects that should be axis-aligned, either axis-align the composite object, or the individual objects, but not both.

Returning to our toy example from section 2, the axis-aligned result is:

```
\entrymodifiers={+!!<0pt,\fontdimen22\textfont2>}
\xymatrix{\toyexample}
```



8 Caveats

8.1 Size

By default, Xy-pic builds objects in `\textstyle`.[4] If, say, `\scriptstyle` is used instead, then each use of `\textfont2` in our solution should be replaced with `\scriptfont2`. After all, the distance between the baseline and axis depends on the size of the math font.

8.2 Labels

By default, Xy-pic places arrow labels halfway between the Xy-pic reference points of the source and destination objects. Even though the shenanigans in this article move the reference point away from the object's center, from the viewpoint of the underlying canvas, it is the object that moves, not the reference point! On the canvas, the final locations of the reference points remain the same, and thus all labels and arrow destinations remain put. However, objects shift vertically, thus affecting their bounding boxes and the *lengths* of arrows. That in turn affects our perception of whether a label is properly placed. In short, hand-tuned code that does a great job with labels on a center-aligned diagram may not do a great job on an axis-aligned diagram. Moral: first settle on a choice of diagram alignment, then tune the placement of your labels.

[4] [XY MANUAL, §4] incorrectly claims the default to be `\displaystyle`. To actually obtain `\displaystyle`, one puts `\objectstyle=\displaystyle`.

9 Proposed enhancements to Xy-pic

Both from the public Xy-pic list

`http://tug.org/mailman/listinfo/xy-pic`

and from private email exchanges, I gather that the authors of Xy-pic welcome the discussion of ideas for improving Xy-pic. Perhaps someone familiar with the source code of Xy-pic could experiment with implementations of the following ideas.

1. The kernel language might support the drop modifier `!A` to have the same effect as

   `!<0pt,\fontdimen22\textfont2>`.

   (The letter 'A' reminds us of "axis" and "alignment".) Actually, the definition should depend on `\objectstyle`. For example, with

   `\objectstyle=\scriptstyle`,

   `!A` should be shorthand for

   `!<0pt,\fontdimen22\scriptfont2>`.

2. The `\xymatrix` feature could support `@A` as a setup to have the same effect as setting

   `\entrymodifiers={+!!A}`.

   Actually, the source code indicates that the default value is `\entrymodifiers={\entrybox}`, and the source for `\entrybox` seems to do more than `\entrymodifiers={+}` would do. Is that true? If so, `@A` should probably also use the more complicated behavior. The point is to gain axis alignment without losing something else.

3. The setup `@1` could be redefined to simply mean `@A@M=1pt`. As discussed in section 3, today's `@1` is a buggy construct: the math strut negatively affects vertical spacing and arrows; in particular, today's `@1` even fails to properly align simple one-line diagrams like $A \twoheadrightarrow A^2$.

4. The matrix option might support some kind of global `\everyxymatrix={...}`, so that one can easily specify setups like `@A` once and for all in a document's preamble. The alternative is to put

`\entrymodifiers={+!!A}` in the preamble, but newcomers to XY-pic are likely to master the use of `\xymatrix` setups prior to tackling kernel-level drop modifiers.

5. Similarly, the graph option might support `\everyxygraph={...}`. When missing, the usual defaults [XY MANUAL, p.53] would apply.

10 Acknowledgments

The problem of aligning objects and arrows appropriately to achieve beautiful diagrams is hardly new. (Just think of TEX itself, or all the diagram packages available on CTAN.) Within the context of XY-pic, the problem was discussed and solved in 2001 on the XY-pic list by Vadim Radionov, whose solution is essentially identical to the one obtained here.

I thank Ross Moore and Florian Lengyel for commenting on earlier versions of this article, and Michael Abbott for discussing the ideas in section 9.

◇ Alexander R. Perlis
Department of Mathematics
The University of Arizona
Tucson, AZ 85721 USA
aprl@math.arizona.edu