Technical notes on equation breaking

Michael J. Downes Morten Høgholm

June 1, 2008

Chapter 1

Tag placement

The method used by the breqn package to place the equation number is rather more complicated than you might think, and the whole reason is to allow the number to stay properly centered on the total height even when the height fluctuates due to stretching or shrinking of the page.

Consider the following equation:

$$N_0 \simeq \left(\frac{\nu}{\|u\|_{H^i}} \right) |I|^{-1/2} \tag{3.15}$$

It will have only one line, if the column width is not too narrow.

Scrutinizing the vertical list will shed light on some of the basic properties shared by all breqn equations. After that we will look at what would happen if two or more lines were needed. The numbers added on the left in the following `\showlists` output mark the points of interest.

```
[1] \penalty 10000
    \glue(\abovedisplayskip) 0.0
    \penalty 10000
    \glue(\belowdisplayskip) 0.0
[2] \glue 4.0 plus 4.0
    \glue(\lineskip) 1.0
[3] \vbox(16.53902+0.0)x0.0, glue set 16.53902fil
    .\glue 0.0 plus 1.0fil minus 1.0fil
    \penalty 10000
[4] \glue -8.51945
[5] \hbox(7.5+2.5)x25.55563
    .\OT1/cmr/m/n/10 (
    .\OT1/cmr/m/n/10 3
    .\OT1/cmr/m/n/10 .
    .\OT1/cmr/m/n/10 1
    .\OT1/cmr/m/n/10 5
    .\kern 0.0
    .\OT1/cmr/m/n/10 )
    \penalty 10000
```

```
[6] \glue(\parskip) -18.01956
[7] \hbox(16.53902+9.50012)x360.0, glue set 1.78647
```

1. These four lines are a hidden display structure from TEX's primitive `$$` mechanism. It is used only to get the value of `\predisplaysize` so that we can later calculate by hand whether to use the short display skips or the regular ones. (The reason that we have to do it by hand traces back to the fact that TEX 3.x does not allow unhboxing in math mode.) The penalties come from `\predisplaypenalty` and `\postdisplaypenalty`, which were locally set to 10000 to ensure there would be no unintended page breaks at these glue nodes.

2. These two glue nodes are the ones that would normally have been produced at the top of a display; the first one is the above-display skip node (though we had to put it in by hand with `\vskip`) and the second one is the usual baselineskip/lineskip node.

3. This is a dummy copy of the equation's first line, which is thrown in here to get the proper value of baselineskip (or lineskip in this case). Why do we need this? Because this ensures that we get the top spacing right before we fiddle with the glue nodes surrounding the equation number. And if the equation has a frame, this box is a good place to add it from.

4. This is a special glue node that brings us to the right vertical position for adding the equation number. Its value is calculated from the variables that you would expect, given the presence of the dummy first line above the num- ber: starting position of the equation, height of first line, total height of equation body. If the equation body had more than one line, with stretchable glue between the lines, half of the stretch would be added in this glue node.

5. The hbox containing the equation number.

6. Backspace to bring the equation body to the right starting point. We use `\parskip` to put this glue in place because we're going to get a parskip node here in any case when we add the equation body with (in essence). If we didn't do this we'd get two glue nodes instead of one, to no purpose.

   `\ \unhbox\EQ@box`.

7. And lastly we see here the first line of the equation body, which appears to have height 16.5pt and depth 9.5pt.

For comparison, the vertical list produced from the above equation in standard LATEX would look like this, if the same values of columnwidth and abovedisplayskip are used:

```
[1] \penalty 10000
[2] \glue(\abovedisplayskip) 4.0 plus 4.0
    \glue(\lineskip) 1.0
    \hbox(16.53902+9.50012)x232.94844
[3] .\hbox(7.5+2.5)x25.55563
    ..\hbox(7.5+2.5)x25.55563
    ...\OT1/cmr/m/n/10 (
    ...\OT1/cmr/m/n/10 3
    ...\OT1/cmr/m/n/10 .
    ...\OT1/cmr/m/n/10 1
```

```
    ...\OT1/cmr/m/n/10 5
    ...\kern 0.0
    ...\OT1/cmr/m/n/10 )
    .\kern101.49591
[4] .\hbox(16.53902+9.50012)x105.8969
    ...
[5] \penalty 0
[6] \glue(\belowdisplayskip) 4.0 plus 4.0
    \glue(\lineskip) 1.0
    \hbox(6.94444+1.94444)x345.0, glue set 62.1106fil
```

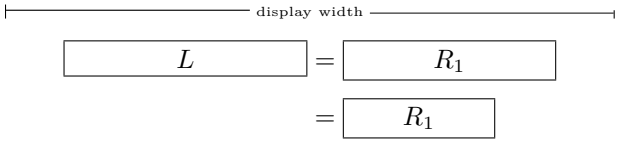
1. `\predisplaypenalty`
2. `\abovedisplayskip`
3. equation number box
4. equation body
5. `\postdisplaypenaltly`
6. `\belowdisplayskip`

Chapter 2

Equation Layouts

2.1 Misc examples

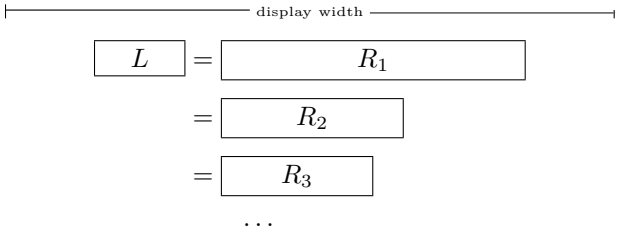
Let us consider which of these have 50% or more of wasted whitespace *within the bounding box of the visible material*.



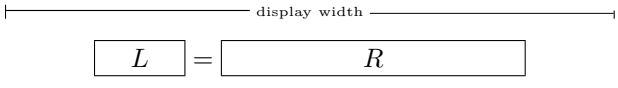
2.2 Ladder and step layouts

2.2.1 Straight ladder layout

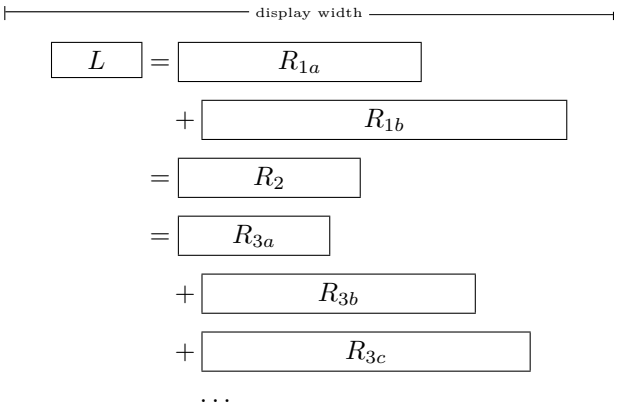
This is distinguished by a relatively short LHS and one or more RHS's of any length.



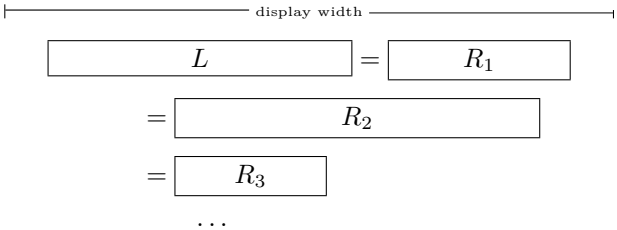
The simplest kind of equation that fits on one line and has only one RHS may be viewed as a trivial subcase of the straight ladder layout:



If some of the RHS's are too wide to fit on a single line they may be broken at binary operator symbols such as plus or minus. This is still classified as a straight ladder layout if none of the fragments intrude into the LHS column, because the underlying parshape is the same.



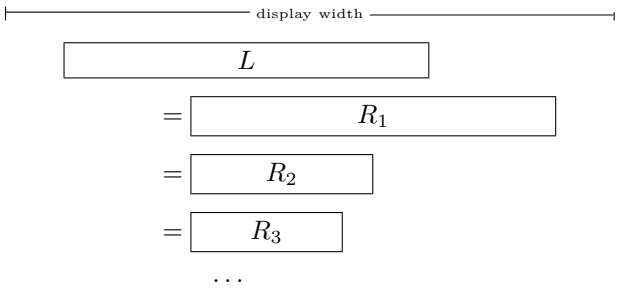
2.2.2 Skew ladder layout



In a skew ladder layout, the combined LHS width plus width of R_1 does not exceed the available width, but one of the other RHS's is so wide that aligning its relation symbol with the others cannot be done without making it run over the right margin:

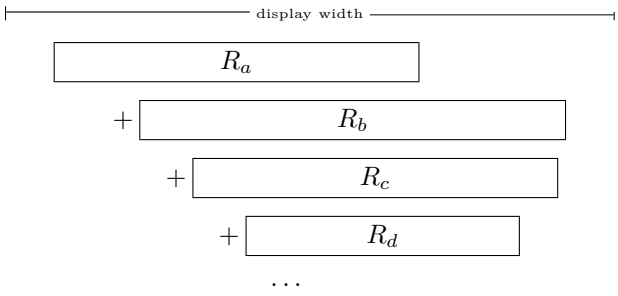
width(L) + width$_{\max}(R_i)$ > width$_{\text{avail}}$. In that case we next try aligning all but the first relation symbol, allowing all the R_i after R_1 to shift leftward.

2.2.3 Drop ladder layout



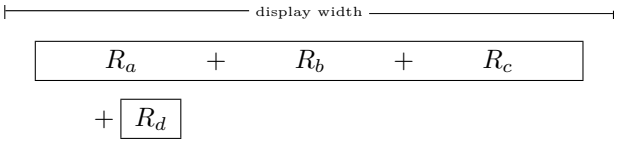
The drop ladder layout is similar to the skew ladder layout but with the width of R_1 too large for it to fit on the same line as the LHS. Then we move R_1 down to a separate line and try again to align all the relation symbols. Note that this layout consumes more vertical space than the skew ladder layout.

2.2.4 Step layout

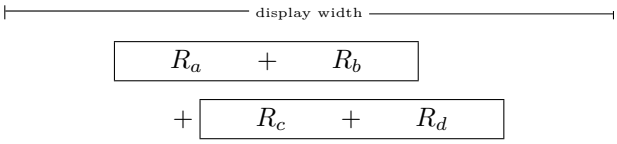


The chief characteristic of the step layout is that there is no relation symbol, so that the available line breaks are (usually) all at binary operator symbols. Let w_1 and w_l be the widths of the first and last fragments. We postulate that the ideal presentation is as follows: Choose a small stairstep indent I (let's say 1 or 2 em). We want the last fragment to be offset at least I from the start of the first fragment, and to end at least I past the end of the first fragment. If there are only two lines these requirements determine a target width $w_T = \max(w_1+I, w_l+I)$. If there are more than two lines ($l > 2$) then use $w_T = \max(w_1 + (l-1)I, w_l + I, w_{\text{avail}}$ and reset I to $w_T/(l-1)$ if $w_T = w_{\text{avail}}$.

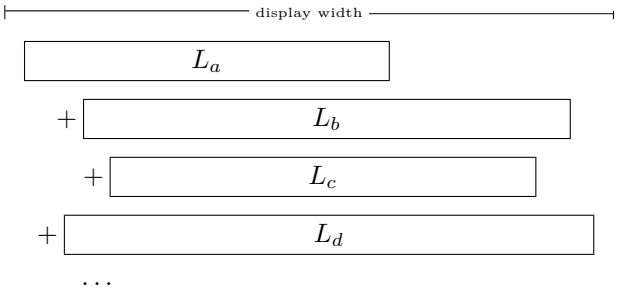
Furthermore, we would like the material to be distributed as evenly as possible over all the lines rather than leave the last line exceedingly short. If the total width is 1.1(width$_{\text{avail}}$), we don't want to have .9 of that on line 1 and .2 of it on line 2:



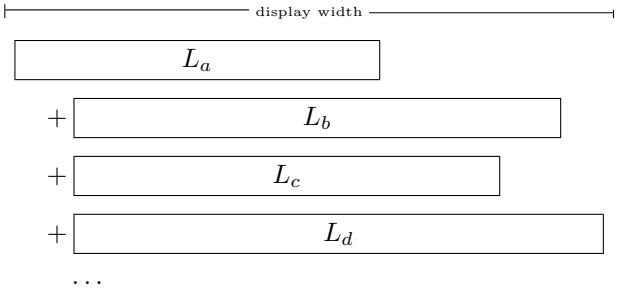
Better to split it as evenly as possible, if the available breakpoints permit.



A degenerate step layout may arise if an unbreakable fragment of the equation is so wide that indenting it to its appointed starting point would cause it to run over the right margin. In that case, we want to shift the fragment leftward just enough to bring it within the right margin:



And then we may want to regularize the indents as in the drop ladder layout. Let's call this a dropped step layout:



2.3 Strategy

Here is the basic procedure for deciding which equation layout to use, before complications like equation numbers and delimiter clearance come into the picture. Let A be the available width, w_{total} the total width of the equation contents, $w(L)$ the width of the left-hand side, $w_{\max}(R)$ the max width of the right-hand sides, I the standard indent for step layout, and O the standard offset for binary operators if a break occurs in the middle of an RHS. Also let t_L and t_R represent certain thresholds for the width of the LHS or the RHS at which a layout decision may change, as explained below.

(1) *Does everything fit on one line?* $w_{\text{total}} \le A$?

Yes: print the equation on a single line (done).

No: Check whether the equation has both LHS and RHS (2).

(2) *Is there a left-hand side?* Are there any relation symbols in the equation?

Yes: Try a ladder layout (3).

No: Try a step layout (10).

(3) *Does the LHS leave room to fit the widest RHS?* $w(L) + w_{\max}(R) < A$?

Yes: Use a straight ladder layout (5).

No: Check the width of the LHS (4).

(4) *Is the LHS relatively short?* $w(L) \le t_L$? (where t_L is typically $0.4A$).

Yes: Subdividing one or more of the RHS's may permit us to use a straight ladder layout (5).

No: The straight ladder layout is unlikely to work. Try a skew or drop ladder layout (6).

(5) *Straight ladder layout* Set up a straight ladder parshape [0pt A $w(L)$ $A-w(L)$] and run a trial break. If the combined width of the LHS plus the longest RHS is no greater than A then we should get a satisfactory layout with all line breaks occurring at major division points (relation symbols). Otherwise, we hope, some additional line breaks at minor division points will allow everything to fit within the text column.

Line breaks OK?

Yes: The straight ladder layout succeeded (done).

No: Try a skew or drop ladder layout (6).

(6) *Do the LHS and the first RHS fit on one line?* $w(L) + w(R_1) \le A$?

Yes: Try a skew ladder layout (7).

No: Try a drop ladder layout (8).

(7) *Skew ladder layout* Set up a parshape [0pt A I $A-I$] and run a trial break.

Line breaks OK?

Yes: Skew ladder layout succeeded (done).

No: One of the unbreakable fragments of the R_i ($i > 1$) is wider than $A-I$; try an almost-columnar layout (9).

(8) *Drop ladder layout* Set up a parshape [0pt $w(L)$ I $A - I$] and run a trial break. This is the same parshape as for a skew ladder layout except that the width of the first line is limited to the LHS width,

so that the RHS is forced to drop down to the next line.

Line breaks OK?

Yes: Drop ladder layout succeeded (done).

No: One of the unbreakable fragments of the R_i ($i > 1$) is wider than $A-I$; try an almost-columnar layout (9).

(9) *Almost-columnar layout* This presupposes a trial break that yielded a series of expressions or fragments, one per line. Let $w(F)$ denote the width of the first fragment and $w(R_i)$ the widths of the remaining fragments. Set up a parshape [0pt $w(F)$ $A - w_{\max}(R_i)$ $w_{\max}(R_i)$]: in other words, set the first line flush left and the longest line flush right and all other lines indented to the same position as the longest line. But as a matter of fact there is one other refinement for extreme cases: if $w_{\max}(R_i) > A$ then the parshape can be simplified without loss to [0pt $w(F)$ 0pt A]—for that is the net effect of substituting $\min(A, w_{\max})$ in stead of $w_{\max}$. (Done.)

(10) *Step layout* Set target width w_T to $A - 2I$. Set parshape to [0pt w_T I $w_T - I$ $2I$ $w_T - 2I$ $\ldots$ $(l-1)I$ $w_T - (l-1)I$], where $l = \lceil w_{\text{total}}/A \rceil$ is the expected number of lines that will be required. Trial break with that parshape in order to find out the width of the last line.

Indents OK?

Yes: Step layout succeeded (done).

No: One of the fragments is too wide to fit in the allotted line width, after subtracting the indent specified by the parshape. Try a dropped step layout (11)

(11) *Dropped step layout* Set up a parshape [0pt A I $A-I$] and run a trial break. Note that this is actually the same parshape as for a skew ladder layout.

Line breaks OK?

Yes: Dropped step layout succeeded (done).

No: One of the unbreakable fragments of the R_i ($i > 1$) is wider than $A-I$; as a last resort try an almost-columnar layout (9).