TEX's "additional demerits" parameters

Udo Wermuth

Abstract

TEX has three integer parameters, `\adjdemerits`, `\finalhyphendemerits`, `\doublehyphendemerits`, which are added to the line demerits by the line-breaking algorithm if certain conditions are met. This article lists which lines of a paragraph can be affected by these parameters and presents some simple techniques to avoid their extensive application in a paragraph.

1 Introduction

The line-breaking procedure of TEX assigns to a set of line breaks for a paragraph a numerical value called its *demerits*; the higher this value the less desirable it is for TEX to use this set of line breaks to typeset the paragraph. The demerits of a paragraph are the sum of the demerits of each line. The calculation of these *line demerits* applies several concepts: The non-negative *badness* of the line, which is based on the width of the white spaces in the line, a penalty value that is added to the badness values, the *penalty* associated with the place at which the line is broken, and visual characteristics involving also the previous line. This last concept is realized by adding the integer values of three parameters, called the *additional demerits*.

The following formula calculates the line demerits, Λ_ι, for line number ι:

$$\Lambda_\iota = (\lambda + \beta_\iota)^2 + \operatorname{sgn}(\pi_\iota)\pi_\iota^2 + \delta_\iota \tag{1}$$

where λ is the `\linepenalty` (which is a constant for a single paragraph), β_ι stands for the badness of the line, π_ι represents the penalty at the line break, and δ_ι is the sum of the applicable additional demerits. Penalties lie in the range $-10000 \le \pi < 10000$ and they keep their sign in (1) although the value is squared. But the value -10000 is ignored in the calculation of the line demerits. A detailed explanation of how TEX breaks paragraphs into lines is given in Chapter 14 of [2]; Section 2 of [6] contains a shorter description that focuses on the parameters involved and the formulas for the calculations.

Traces. TEX displays most of the values that are used in formula (1) if `\tracingparagraphs` is positive. In this case the log file of the run contains trace data written by the line-breaking algorithm. Two types of lines of this trace are important to understand this article:

Break candidates signal a valid way to break a line. Their trace lines start with a single `@` and they contain the values for β_ι, π_ι, and Λ_ι. The line number ι is not yet known. For example,

```
@\par via @@1 b=1 p=50 d=7621    (*)
```

gives $\beta_\iota = 1$, $\pi_\iota = 50$, and $\Lambda_\iota = 7621$.

Feasible breakpoints, which follow one or more break candidates, show the best way to break the text up to this point. Their trace lines start with `@@` and contain a *sequence number*, the line number, and a *fitness class*. For example, in

```
@@2: line 1.3- t=3125 -> @@0    (**)
```

the sequence number is 2, the line number $\iota = 1$, and the fitness class 3. The reference `-> @@0` shows that the previous feasible breakpoint for this breakpoint has the sequence number 0.

The sequence numbers are needed to link the feasible breakpoints to determine the above mentioned set of line breaks; the sequence number 0 is used for the start of the paragraph. The fitness class is a qualitative indicator of the way the white space in the line is altered. There are four classes: *very loose*, *loose*, *decent*, and *tight* which carry the numbers 0 to 3, resp., in the trace line of type (∗∗). See pages 98–99 of [2] or Section 3 of [6] for a detailed description of the trace data.

Note that the additional demerits, δ_ι, are not listed in the trace data. They must be computed from the given values using equation (1) in the following form:

$$\delta_\iota = \Lambda_\iota - (\lambda + \beta_\iota)^2 - \operatorname{sgn}(\pi_\iota)\pi_\iota^2. \tag{2}$$

Plain TEX sets $\lambda = 10$ [2, p. 98], so Λ_ι in (∗) contains the following amount of additional demerits:

$$\delta_\iota = 7621 - (10 + 1)^2 - {+50}^2 = 5000.$$

Paths. Starting with the last feasible breakpoint the line breaks can be found by going back through the sequence of the linked previous feasible breakpoints. If all feasible breakpoints appear in this sequence and all of them have only one associated break candidate, TEX has only one way to typeset the paragraph. But often several break candidates are followed by several feasible breakpoints and then many possibilities for the next breakpoint are available. TEX chooses — obeying the current setting of `\looseness` — the set of line breaks, i.e., feasible breakpoints, that has the lowest sum of line demerits, also called the *total demerits* of the paragraph. (The current total demerits at a feasible breakpoint is shown after the `t=` in lines of type (∗∗).) Each set for the whole paragraph is called a *path* and its sum of line demerits *path demerits*, Λ_p. The entirety of all paths form a *network* (see [1, Fig. 13]).

Traces can get very long when TeX finds a lot of feasible breakpoints, and then they document a lot of paths in the network. Such paths are difficult to compare when only the trace data is available. This article shows only short traces and switches to a table form for longer ones. This table form is introduced and described in Section 5 of [7]. Both TeX's trace data and the table form are also briefly explained in this article.

Additional demerits. The three additional demerit parameters fall into two groups: Two deal with hyphens at the end of lines, the third with the visual appearance of lines. The following three integer parameters hold the values of the additional demerits.

`\finalhyphendemerits` is used in the last line if the penultimate line of the paragraph ends in a hyphen.

`\doublehyphendemerits` is applied to the second line of a pair of lines if both lines end in a hyphen. But it is not applied to the last line of the paragraph.

`\adjdemerits` is charged to the second line of a pair of lines if they are *visually incompatible*. This means that the glue in the two lines is set quite differently: In one line it is extremely stretched compared to the other. (The first line of a paragraph is compared to a line with decent spacing.) In TeX's view the fitness class numbers in linked feasible breakpoints must differ by more than 1 to add the value of this parameter.

Plain TeX assigns the values 5000, 10000, and 10000 to the three parameters [2, p. 98], resp. The values of these parameters are named in this article δ_f, δ_d, and δ_a, resp. Note that δ_d and δ_f are never applied together in a line.

Implementation. In the section "More Bells and Whistles" of [1] it is written that the parameter `\adjdemerits` was added — with a high price paid in the implementation through the introduction of the fitness classes — as a more or less experimental feature. The design and implementation was done in the spring of 1980 (p. 143 of [5]); Chapter 11 in [4] shows it as entry #461. As it is available today it seems to be worth the price. Note that only the parameter `\adjdemerits` is likely to be used in the first pass as hyphenated lines must use a (typically rare) author-entered hyphen in this pass.

In the implementation of the line-breaking algorithm the values of the integer parameters that represent the additional demerits are just one summand in the calculation of the line demerits [3, §859]. But the implementation uses `\adjdemerits` in a second place to keep the number of feasible breakpoints and break candidates small [3, §836]. This means changing the values of these integer parameters might result in a different number of lines in the trace.

Of course, a path in the network created from TeX's default settings exists also as a valid set of line breaks with changed values of the parameters. But TeX does not follow a path to its end as soon as it learns that this path cannot lead to the minimal total demerits. The trace data often shows only the beginning of paths. In the table form of the trace data these partial paths are extended to a complete path using the available feasible breakpoints; such paths are described as "hidden in the trace data".

Usage. The three parameters try to prevent TeX from picking a path that contains undesirable constructions. The techniques of this article might turn out to be useful for an author to handle situations in which TeX chose unwanted line breaks. But almost always it is better to rewrite a paragraph if it looks bad than to adjust TeX's line-breaking parameters.

In the following sections the three parameters are discussed one by one. Note, however, that a change to the parameters affects all following paragraphs so that a changed value should be used inside a group around a paragraph, which must end inside this group, i.e., write an empty line or `\par` before ending the group. TeX calculates with the values assigned to the parameters that it knows at the end of a paragraph.

2 About `\finalhyphendemerits`

The effect of `\finalhyphendemerits` is probably easier to understand than the effects of the other two parameters. TeX charges δ_f to the last line of a path if and only if its penultimate line ends in a hyphen or a ligature in which the last character is a hyphen; in the `cm` fonts the line must end with a hyphen, an en-dash, or an em-dash. TeX has to look at only a single line in contrast to the other two additional demerits where TeX judges about pairs of lines.

Example 1: Description
Typeset a paragraph whose penultimate line ends in an em-dash.

TeX input

```
The em-dash at the end of this line---this
one---adds {\tt\char92finalhyphendemerits\/}
to the last line.\par
```

TeX output

The em-dash at the end of this line—this one—
adds `\finalhyphendemerits` to the last line. ▯

(Note: The symbol '▯' marks the end of an example.)

With `\tracingparagraphs` = 1 trace lines appear in the log file — here formatted to fit the column width and with line numbers for reference.

Example 1 continued: Log file contents

```
 1. @firstpass
 2. []\tenrm The em-dash at the end of this
      line---this one---
 3. @\discretionary via @@0 b=11 p=50 d=2941
 4. @@1: line 1.2- t=2941 -> @@0
 5. adds \tentt \finalhyphendemerits \tenrm to
      the last line.
 6. @\par via @@1 b=0 p=-10000 d=5100
 7. @@2: line 2.2- t=8041 -> @@1
```
□

Lines 3 and 6 contain break candidates, lines 4 and 7 feasible breakpoints. As there is only one break candidate per feasible breakpoint TeX has exactly one path to typeset this paragraph. Lines 3 and 4 show a penalty for the first line; it states that the `\exhyphenpenalty` is applied, i.e., the em-dash is considered an author-entered hyphen. (Plain TeX sets `\hyphenpenalty` and `\exhyphenpenalty` to 50 [2, p. 96].) The calculation using (2) shows for this break candidate representing output line 1:

$$\delta_1 = 2941-(10+11)^2-50^2 = 2941-441-2500 = 0.$$

The badness of the last line is 0; see trace lines 6 (and 7). As mentioned above a penalty of −10000 is ignored in the calculation of the line demerits. Therefore (2) gives for the second line:

$$\delta_2 = 5100-(10+0)^2 = 5100-100 = 5000.$$

Thus, as expected, TeX has applied the value δ_f to the last line.

Changing this parameter. If δ_f is applied to a paragraph and if TeX's line-breaking network contains at least one path that avoids a hyphen at the end of the second-last line a high enough value for `\finalhyphendemerits` will select such a path.

Example 2: Description

Typeset a paragraph twice: First with TeX's default values and a second time with a larger value of `\finalhyphendemerits`.

TeX input

```
\noindent In this case \TeX's network shows
2 paths for its line-breaking decision.\par
```

TeX output

In this case TeX's network shows 2 paths for its line-breaking decision. □

This is the trace data:

Example 2 continued: Log file contents

```
 1. @firstpass
 2. \tenrm In this case T[]X's network shows 2
      paths for its
 3. @ via @@0 b=96 p=0 d=11236
 4. @@1: line 1.1 t=11236 -> @@0
 5. line-
 6. @\discretionary via @@0 b=29 p=50 d=4021
 7. @@2: line 1.3- t=4021 -> @@0
 8. breaking decision.
 9. @\par via @@1 b=0 p=-10000 d=100
10. @\par via @@2 b=0 p=-10000 d=5100
11. @@3: line 2.2- t=9121 -> @@2
```
□

The feasible breakpoint `@@3` has two break candidates and TeX has a choice: Go via `@@2` as suggested in `@@3` or use the break candidate in line 9 without the application of δ_f and link `@@3` to `@@1`.

To select the path that uses the feasible breakpoints `@@1` and `@@3` and thus avoids the hyphen in the penultimate line the total demerits from the path via `@@2` and `@@3`, $\Lambda_{\mathrm{p}[2,3]}$, must become larger than the path demerits of the requested path, $\Lambda_{\mathrm{p}[1,3]}$. This can be done by increasing the value of δ_f. Let's mark its new value with a prime, then the inequality

$$\Lambda_{\mathrm{p}[2,3]} + \delta'_f - \delta_f > \Lambda_{\mathrm{p}[1,3]}$$

must hold. Simple transformations give

$$\begin{aligned} & \delta'_f > \Lambda_{\mathrm{p}[1,3]} - \Lambda_{\mathrm{p}[2,3]} + \delta_f \\ \Longrightarrow\quad & \delta'_f > (11236+100) - 9121 + 5000 = 7215 \end{aligned}$$

and the value for `\finalhyphendemerits` should be at least 7216. The first line then becomes loose with a badness of 96 (see lines 3 and 4) instead of tight with a badness of 29 (see lines 6 and 7). It is a trade-off: To avoid the hyphen in the second-last line TeX has to select a worse solution at other places. Here it is a loose line.

Example 2 continued: TeX input

```
\finalhyphendemerits=7216
\noindent In this case \TeX's network ...
```

TeX output

In this case TeX's network shows 2 paths for its line-breaking decision. □

Of course, it is not necessary to calculate the minimal δ_f; it can be set to a high value like 7500 or 10000 or 100000 to get the desired result.

3 About `\doublehyphendemerits`

The parameter `\doublehyphendemerits` is applied to the second line of a pair of lines if both lines end in a hyphen and the second line is not the last line. This restriction comes from the fact that TeX treats the end of a paragraph as if it ends in a hyphen [3, §829]. An author-entered hyphen at the end of a paragraph does not make a difference. (It is rare indeed to have a hyphen at the end of a paragraph, so it is not an important case.) Thus, in a paragraph all lines except the first and the last might be charged with δ_d, the value of this parameter.

Both explicit hyphens, which are entered by the author, and implicit hyphens, which are inserted by TeX's hyphenation algorithm, are treated in the same way by TeX when it determines if a line ends in a hyphen.

Example 3: Description

Typeset a paragraph with a lot of hyphenated lines.

TeX input

```
In this paragraph \TeX\ must apply its
hyphenation procedure. And the last line ends
in an em-dash. Parameter
{\tt\char92doublehyphendemerits\/} is
applied twice in four lines---\par
```

TeX output

In this paragraph TeX must apply its hyphenation procedure. And the last line ends in an em-dash. Parameter `\doublehyphendemerits` is applied twice in four lines— □

As always, useful information is found in the trace data; the first line states that TeX uses the second pass (lines from the first pass are not shown).

Example 3 continued: Log file contents

```
 1. @secondpass
 2. []\tenrm In this para-graph T[]X must
      ap-ply its hy-phen-
 3. @\discretionary via @@0 b=0 p=50 d=2600
 4. @@1: line 1.2- t=2600 -> @@0
 5. ation pro-ce-dure. And the last line ends
      in an em-
 6. @\discretionary via @@1 b=7 p=50 d=12789
 7. @@2: line 2.2- t=15389 -> @@1
 8. dash. Pa-ram-e-ter \tentt
      \doublehyphendemerits \tenrm is ap-
 9. @\discretionary via @@2 b=147 p=50 d=47149
10. @@3: line 3.0- t=62538 -> @@2
11. plied twice in four lines---
12. @\par via @@3 b=0 p=-10000 d=*
13. @@4: line 4.2- t=62538 -> @@3
```
□

Lines 6 and 9 show that δ_d was applied. In line 12 TeX has not calculated the demerits and sets them to 0; compare the values after `t=` in lines 10 and 13 of the trace. TeX does not calculate the demerits as the necessary feasible breakpoint at the end of the paragraph follows a possible break after the em-dash (see [3, §854]).

Remove a stack of hyphens. Note that there is no way to prevent the three consecutive hyphenated lines in example 3: TeX has no path in its line-breaking network to avoid this stack of hyphens. But if there is such a path a change of the value of `\doublehyphendemerits` can trade in higher badness values and/or more penalties and/or more visually incompatible lines and/or more lines in the paragraph to avoid such a stack of hyphens.

Table 1: Badness, penalties, and additional demerits of the line breaks for the paths of example 4

		`\par via @@` (`*` is typeset)				
`@@`	Class	$*7_0$	8_0	7_1	7_2	8_1
1	d	4_{50}	4_{50}	4_{50}	4_{50}	4_{50}
2	v				118^{a}	
3	d	0^{d}_{50}	0^{d}_{50}	0^{d}_{50}		0^{d}_{50}
4	l			51^{d}_{50}		
(4)	(d)				1^{a}_{50}	
5	d	0^{d}_{50}				0^{d}_{50}
6	t		79			
7	l	33				
(7)	(t)			15^{a}	15	
8	l		17^{a}_{50}			
(8)	(t)					98^{d}_{50}
9	d	0	0^{f}	0	0	0^{f}
# lines =		5	5	5	5	5
$\sum$ badness =		37	100	70	138	102
# a/d/f =		0/2/0	1/1/1	1/2/0	2/0/0	0/3/1
$\Lambda_{\mathrm{p}}(10)$ =		29845	41546	42242	42426	57160

Example 4: Description

Typeset a paragraph twice: First with TeX's default values and a second time with a larger value of `\doublehyphendemerits`.

TeX input

```
Final remark: Values for the badness are
sometimes stated as {\tt*} which means that it
is {\sl infinite\/} according to \TeX's rules.
For demerits such an asterisk means that the
calculation was not performed because of
certain forced conditions.\par
```

TeX output

Final remark: Values for the badness are sometimes stated as `*` which means that it is *infinite* according to TeX's rules. For demerits such an asterisk means that the calculation was not performed because of certain forced conditions. □

This time the data of the trace is shown in a more compact form as there are several paths that must be studied.

Table 1 summarizes the paths that can be identified in the trace data. The table shows in the first two columns the information of the `@@`-lines, i.e., the feasible breakpoints: the sequence number and the fitness class abbreviated to the first letter of very loose, loose, decent, or tight. The next two columns present the explicitly shown `\par`-lines in the trace (like line 12 in the trace of example 3). The heading gives the sequence number after the "`via @@`" marked with a subscript 0. The next three columns are "hidden" paths in the trace that are not followed to their end by TeX; such paths get subscripts larger than 0. As these paths use `@@`-alternatives that are not shown in the trace some sequence numbers and their fitness classes are placed in parentheses. The column head of 7_0 contains an asterisk to indicate that TEX has selected this path for typesetting.

The table entries are the badness values. A subscript signals that a penalty with the stated value occurs at the break, a superscript of 'f', 'd', or 'a' that δ_f, δ_d, or δ_a, resp., is applied.

The last four rows state the number of lines, the sum of the badness values of the path, the number of lines with δ_a, δ_d, δ_f, and the path demerits $\Lambda_{\mathrm{p}}(10)$ with the value 10 for `\linepenalty`. Most of these values are not found in the trace data; they have been computed from the information in the columns.

Column 7_0 (with the path that TEX uses) contains two applications of δ_d but the path in column 7_2 none. Therefore, it is possible to typeset the paragraph without three consecutive hyphens if the value δ_d is changed. If the new value is named δ'_d then

$$\Lambda_{\mathrm{p}[7_0]} + 2\delta'_d - 2\delta_d > \Lambda_{\mathrm{p}[7_2]}$$

must hold. This means, the path demerits of the path 7_0, which represent currently the total demerits of the paragraph, must get larger than the path demerits of path 7_2. The calculation is

$$\begin{aligned} & \delta'_d > (\Lambda_{\mathrm{p}[7_2]} - \Lambda_{\mathrm{p}[7_0]} + 2\delta_d)/2 \\ \Longrightarrow \quad & \delta'_d > (42426 - 29845 + 20000)/2 = 16290.5. \end{aligned}$$

Example 4 continued: TEX input

```
\doublehyphendemerits=16291
Final remark: Values for the badness are ...
```

TEX output

Final remark: Values for the badness are sometimes stated as * which means that it is *infinite* according to TEX's rules. For demerits such an asterisk means that the calculation was not performed because of certain forced conditions. □

The same technique can be tried to make TEX select the path of column 8_0. This time both sides list δ'_d:

$$\Lambda_{\mathrm{p}[7_0]} + 2\delta'_d - 2\delta_d > \Lambda_{\mathrm{p}[8_0]} + \delta'_d - \delta_d.$$

Therefore

$$\begin{aligned} & \delta'_d > \Lambda_{\mathrm{p}[8_0]} - \Lambda_{\mathrm{p}[7_0]} + \delta_d \\ \Longrightarrow \quad & \delta'_d > 41546 - 29845 + 10000 = 21701. \end{aligned}$$

The computed value 21702 is larger than 16291 but the latter value is sufficient to typeset path 7_2: TEX's second-best path with its default settings, path 8_0, cannot be reached from 7_0 by increasing δ_d.

Changing the other parameters too. Instead of changing only the parameter that is assigned to this kind of trouble, the other parameters can also be involved to select a specific path. To select path 7_2 in example 4 it is not necessary but in other cases it is the only way to make TEX typeset the desired path.

In the inequality to go from path 7_0 to path 7_2, only δ_d was changed. But as 7_2 applies δ_a, the inequality should better be written as

$$\Lambda_{\mathrm{p}[7_0]} + 2\delta'_d - 2\delta_d > \Lambda_{\mathrm{p}[7_2]} + 2\delta'_a - 2\delta_a.$$

This gives an inequality for the difference $\delta'_d - \delta'_a$ that must at least hold to typeset 7_2:

$$\delta'_d - \delta'_a > (\Lambda_{\mathrm{p}[7_2]} - \Lambda_{\mathrm{p}[7_0]})/2 + \delta_d - \delta_a = 6290.5.$$

That is, $\delta'_d = 10000$ and $\delta'_a = 3709$ typeset path 7_2 as well as $\delta'_d = 13300$ and $\delta'_a = 7000$.

Similar, to prefer path 8_0 to 7_0 the inequality is

$$\begin{aligned} \Lambda_{\mathrm{p}[7_0]} + 2\delta'_d - 2\delta_d > \Lambda_{\mathrm{p}[8_0]} &+ \delta'_a + \delta'_d + \delta'_f \\ &- \delta_a - \delta_d - \delta_f \\ \Longrightarrow \quad \delta'_d - \delta'_a - \delta'_f > \Lambda_{\mathrm{p}[8_0]} &- \Lambda_{\mathrm{p}[7_0]} + \delta_d - \delta_a - \delta_f \\ &= 6701. \end{aligned}$$

To get from 7_2 to 8_0, i.e., to avoid that 7_2 is chosen by TEX instead of 8_0, the parameters must also fulfill

$$\begin{aligned} \Lambda_{\mathrm{p}[7_2]} + 2\delta'_a - 2\delta_a > \Lambda_{\mathrm{p}[8_0]} &+ \delta'_a + \delta'_d + \delta'_f \\ &- \delta_a - \delta_d - \delta_f \\ \Longrightarrow \quad \delta'_a - \delta'_d - \delta'_f > \Lambda_{\mathrm{p}[8_0]} &- \Lambda_{\mathrm{p}[7_2]} + \delta_a - \delta_d - \delta_f \\ &= -5580. \end{aligned}$$

The two inequalities give the criteria for how to change the parameters to typeset path 8_0 instead of path 7_0. For example, $\delta'_d = 13300$, $\delta'_a = 7000$, and $\delta'_f = -750$ is one solution. (The additional demerits are integers, so they can receive negative values.)

Example 4 continued: TEX input

```
\doublehyphendemerits=13300 \adjdemerits=7000
\finalhyphendemerits=-750
Final remark: Values for the badness are ...
```

TEX output

Final remark: Values for the badness are sometimes stated as * which means that it is *infinite* according to TEX's rules. For demerits such an asterisk means that the calculation was not performed because of certain forced conditions. □

But it is much simpler to set $\delta'_f = -6702$ in order to make TEX typeset path 8_0. A single change of δ_f helps to avoid the problem with a stack of hyphens without touching δ_d or involving path 7_2. The change suggests to TEX to trade a tolerable hyphen in the penultimate line against the unwanted hyphen stack. For such a trade the integer parameter `\finalhyphendemerits` is ideal as it applies to only one line.

4 About `\adjdemerits`

The integer parameter `\adjdemerits` is applied to the second line of a pair of lines if they are visually incompatible, i.e., if their fitness classes are not adjacent in the sequence *very loose*, *loose*, *decent*, and *tight*. The first line of a paragraph is compared to the fitness class *decent*. That means that a very loose first line receives the additional demerits δ_a.

Example 5: Description
Typeset a paragraph in which each line is charged with `\adjdemerits`.

TeX input

```
So it is a v-loose line, oh, this line must be
{\sl very~loose\/} to charge the famous
``adjacent demerits'' in a {\sl first line\/}
and then follows a {\sl tight\/} line but it
forces that the very next line is {\sl loose\/}
again to charge these adjacent demerits a
third time; and then a repetition of this
tight/loose pattern makes the rest.\par
```

TeX output

So it is a v-loose line, oh, this line must be *very loose* to charge the famous “adjacent demerits” in a *first line* and then follows a *tight* line but it forces that the very next line is *loose* again to charge these adjacent demerits a third time; and then a repetition of this tight/loose pattern makes the rest.□

The log file contains the following trace data:

Example 5 continued: Log file contents

```
 1. @firstpass
 2. []\tenrm So it is a v-loose line, oh, this
     line must be
 3. @ via @@0 b=100 p=0 d=22100
 4. @@1: line 1.0 t=22100 -> @@0
 5. \tensl very loose \tenrm to charge the
     famous ``adjacent demerits''
 6. @ via @@1 b=22 p=0 d=11024
 7. @@2: line 2.3 t=33124 -> @@1
 8. in a \tensl first line \tenrm and then
     follows a \tensl tight \tenrm line but it
 9. @ via @@2 b=37 p=0 d=12209
10. @@3: line 3.1 t=45333 -> @@2
11. forces that the very next line is \tensl
     loose \tenrm again to charge
12. @ via @@3 b=34 p=0 d=11936
13. @@4: line 4.3 t=57269 -> @@3
14. these adjacent demerits a third time; and
     then a
15. @ via @@4 b=70 p=0 d=16400
16. @@5: line 5.1 t=73669 -> @@4
17. repetition of this tight/loose pattern
     makes the rest.
18. @\par via @@5 b=16 p=-10000 d=10676
19. @@6: line 6.3- t=84345 -> @@5
```
□

That TeX assigns to every line the value of `\adjdemerits` can happen only if the number of lines in the paragraph is even. The first line must be *very loose* and all other odd-numbered lines at least *loose*; all even-numbered lines are *tight* or *decent*. As the last line of a paragraph cannot be *loose* with the default setting of `\parfillskip` in plain TeX an odd number of lines in the paragraph is not able to charge `\adjdemerits` either to the last line or to the first if the first line is not very loose.

The trace shows that TeX has no other path to typeset the text. It has no material to trade in order to reduce the number of lines that are charged with δ_a. As there are tight lines the author can help TeX by typing a discretionary hyphen, for example, by entering ``` ``adjacent demer\-its'' ```, which yields:

Example 5 continued: TeX output

So it is a v-loose line, oh, this line must be *very loose* to charge the famous “adjacent demerits” in a *first line* and then follows a *tight* line but it forces that the very next line is *loose* again to charge these adjacent demerits a third time; and then a repetition of this tight/loose pattern makes the rest. □

The paragraph has now a hyphenated line and is one line longer but, except for the unchanged first line, no line is charged with δ_a.

Increase the number of paths. The inserted hyphenation point shows a way that TeX might be able, without any help, to avoid at least some of the visually incompatible lines: allow hyphenation. More variation for line breaks is available if the second pass is forced. Then hyphenation with penalties and the other two additional demerits might appear.

Example 6: TeX input (Example 5 continued)

```
\pretolerance=-1 % suppress the first pass
So it is a v-loose line, oh, this ...
```

TeX output

So it is a v-loose line, oh, this line must be *very loose* to charge the famous “adjacent demerits” in a *first line* and then follows a *tight* line but it forces that the very next line is *loose* again to charge these adjacent demerits a third time; and then a repetition of this tight/loose pattern makes the rest. □

In the second pass the trace gets much longer so that the paths are best presented in the table form: see Table 2. The path shown in column 9_1 is used if the text is typeset in the first pass. In the second pass this path is not followed to its end by TeX; it has many more path demerits than any other set of line breaks. The path of column 11_0 represents the line breaks with the inserted discretionary hyphen.

But TeX finds a third solution. Table 2 shows that the chosen path starts with stretched white spaces in the first two lines and after a decent line the white space has to shrink in two lines. The path

Table 2: Badness, penalties, and additional demerits of the line breaks for the paths of example 6

		`\par` via `@@` (`*` is typeset)				
`@@`	Class	9_0	$*10_0$	11_0	9_1	10_1
1	v	100^a	100^a	100^a	100^a	100^a
2	l	27_{50}	27_{50}	27_{50}		
3	t				22^a	22^a
4	d			1		
5	d	8	8			
(5)	(l)				37^a	37^a
6	l			55		
7	t	34	34		34^a	34^a
8	l			95		
9	l	70^a			70^a	
10	t		56_{50}			56_{50}
11	d			5		
12	d		0^f	0		0^f
(12)	(t)	16^a			16^a	
	# lines =	6	6	7	6	6
	$\sum$ badness =	255	225	283	279	249
	# a/d/f =	3/0/0	1/0/1	1/0/0	6/0/0	4/0/1
	$\Lambda_p(10)$ =	55305	40185	41665	84345	69225

of `11`$_0$ typesets the whole paragraph without tight lines and its penultimate line does not end in a hyphen. Therefore in the forced second pass the setting `\finalhyphendemerits` $= 6481 > 41665 - 40185 + 5000$ suffices to get the path of column `11`$_0$.

5 Summary

It is no surprise that in a paragraph with a hyphen at the end of the penultimate line a larger value of `\finalhyphendemerits`, δ_f, can make TEX select different line breaks which avoid this hyphen. Neither is it a surprise that a larger value of the parameter `\doublehyphendemerits`, δ_d, might prevent TEX from typesetting several consecutive lines that all end in a hyphen nor that a large enough value of `\adjdemerits`, δ_a, might reduce the number of visually incompatible lines in a paragraph.

But to change only one parameter is not always the best solution. For example, if a stack of three hyphens can be avoided by a path with two stacks of two hyphens this path gets the same amount of additional demerits as the original path if only δ_d is changed so TEX will never select this path. Only a path with a smaller number of lines charging δ_d will be eventually considered by TEX.

This article shows that it is not too difficult to determine if the problematic situation can be resolved at all: The trace data that is written to the log file if `\tracingparagraphs` = 1 must contain choices for TEX for the feasible breakpoints of the unwanted lines or the following line. On the other hand the creation of the path tables involves manual work. So these tables are not created ad hoc and are rarely used in typesetting projects.

As the value of `\finalhyphendemerits` is applied to only one line, a new value does not change the order induced by the path demerits for paths that apply δ_f. In this case TEX has always to select the one which has the smallest path demerits with the default value of this parameter. Therefore a low enough negative value makes TEX pick this path: In a paragraph with an unwanted construction but without a hyphenated penultimate line the setting `\finalhyphendemerits` $= -10000$ (or smaller) can be tried. Or first the trace could be checked as the existence of paths with an application of δ_f is readily spotted. The result might not be one of the best solutions but it is easy to produce and at least it offers TEX the ability to trade the hyphen in the second-last line with one application of δ_a or δ_d.

A forced second pass should be executed if a problem with `\adjdemerits` occurs in the first pass before other techniques are tried.

References

[1] Donald E. Knuth and Michael F. Plass, "Breaking paragraphs into lines", *Software — Practice and Experience* **11** (1981), 1119–1184; reprinted with an addendum as Chapter 3 in [5], 67–155.

[2] Donald E. Knuth, *The TEXbook*, Volume A of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1984.

[3] Donald E. Knuth, *TEX: The Program*, Volume B of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1986.

[4] Donald E. Knuth, *Literate Programming*, Stanford, California: Center for the Study of Language and Information, CSLI Lecture Notes No. 27, 1992.

[5] Donald E. Knuth, *Digital Typography*, Stanford, California: Center for the Study of Language and Information, CSLI Lecture Notes No. 78, 1999.

[6] Udo Wermuth, "Tracing paragraphs", *TUGboat* **37**:3 (2016), 358–373; Errata in [7], 414. `tug.org/TUGboat/tb37-3/tb117wermuth.pdf`

[7] Udo Wermuth, "A note on `\linepenalty`", *TUGboat* **38**:3 (2017), 400–414. `tug.org/TUGboat/tb38-3/tb120wermuth.pdf`

◇ Udo Wermuth
Dietzenbach, Germany
`u dot wermuth (at) icloud dot com`

Errata for a previous article. In Table 1′ of [7] the column `12`$_1$ should have 0^a in the row for feasible breakpoint 17 as the previous line is very loose. This adds 10000 demerits to $\Lambda_p(10)$ in this column. The values -139 and -82 in the column of `12`$_1$ of the diagonal matrix after (18) become -207 and -164, resp.