A note on \linepenalty

Udo Wermuth

Abstract

This article analyzes the effect of the line-breaking parameter \linepenalty. First, its rôle in the problem of typesetting a text in one line or in two lines is studied theoretically. Then the effect of different values for \linepenalty are demonstrated for longer paragraphs. Finally, \linepenalty is compared to \looseness.

1 Introduction

The line-breaking algorithm of TEX selects a shortest path in a network of possible breakpoints ([2] or the reprint in [7], p. 107) using a cost function that calculates *demerits*. For every line, four values are involved in the computation of its demerits and then the sum of all line demerits stands for the *total demerits* of a paragraph. Three of the four values are directly related to the characteristics of the created line, while the fourth value is a constant for all lines of a paragraph: the \linepenalty. In TEX78 this constant was hard-coded into the program but with TEX82 it became an integer parameter that the user can change ([5], or the reprint in [6], pp. 273–274). The plain format sets the default value of \linepenalty.

The next section gives a brief overview of the rules by which TEX's line-breaking algorithm calculates the demerits. It also introduces some notation in accordance with [9] but as there are only a few symbols in this article the conventions are not repeated here. Section 3 analyzes what happens to a text, for example, a heading, that can be typeset either in one line or in two lines. Even this simple case gets rather complex and Section 4 summarizes some of the theoretical results and applies them to normal text. The fifth section looks at longer paragraphs and finds some insights when the value of \linepenalty is changed. Section 6 compares the effects of the two integer parameters \linepenalty and \looseness.

The phrases "(possible) solution" or "path in the network", etc., refer to the network of line breaks [2, Fig. 13] that exists for the given text. They do not mean either that this is the shortest path in the network and thus the typeset solution of the line-breaking problem, or that the path is part of the network which is actually created by TEX's line-breaking algorithm. Usually, this algorithm removes the initial segment of a path and thus the path from its memory as soon as it learns that this beginning cannot lead to the shortest path. The phrases here state only that a certain path exists in the whole network.

2 Calculation of demerits

Section 2 of [8] contains a detailed description of the rules that TEX uses to compute the demerits. In order to introduce the notation for this article a brief summary of these rules follows.

The formula ([3], p. 98) that computes the demerits of a line, stated as Λ, is

$$\Lambda = (\lambda + \beta)^2 + \operatorname{sgn}(\pi)\pi^2 + \delta \qquad (1)$$

which names the four parameters with Greek letters.

λ is the \linepenalty, a constant value set in the plain TEX format to 10.

The badness assigned to the line is called β. The badness is itself the result of a computation which looks at the ratio of used and available stretch- or shrinkability in the line ([3], p. 97). It is a nonnegative number $\leq$ \pretolerance in the first pass; otherwise $\leq$ \tolerance.

Depending on the type of line break a penalty π is charged for the break ([3], p. 96). The value is squared but the sign is kept so that the line demerits are lowered when a negative penalty is given. Penalties lie in the range $-10000 < \pi \leq 10000$; $\pi = -10000$ forces a break but does not add to the line demerits. A break at glue gets $\pi = 0$; otherwise a break at a hyphen uses either \hyphenpenalty or \exhyphenpenalty, a break in math mode either \binoppenalty or \relpenalty, and a break at an explicit \penalty command uses the given value. The plain TEX format sets the value of the four mentioned parameters to 50, 50, 700, and 500, respectively.

The last parameter is named the *additional demerits* δ. Lines interact with their predecessors: If visually incompatible lines would be output or if two hyphens in a row occur or if the penultimate line of the paragraph ends with a hyphen then additional demerits are added. The term δ is the sum of the parameters for these three mentioned cases: \adjdemerits, \doublehyphendemerits, and only for the last line \finalhyphendemerits. The default values in plain TEX are 10000, 10000, and 5000, respectively.

The Pascal code in [4, §859] shows that the first summand on the right hand side of equation (1) takes a minimum of two numbers and involves the absolute value of $\lambda + \beta$. Actually the summand is

coded as

$$\big(\min(10000, |\lambda+\beta|)\big)^2.$$

Of course, in the plain TEX format $\lambda+\beta<10000$; but the code states that the value

$$10000-\max(\texttt{\textbackslash pretolerance}, \texttt{\textbackslash tolerance})-1 \quad (2)$$

builds an upper limit for a positive `\linepenalty`.

If the line demerits are calculated for line number ι then Λ, β, π, and δ receive ι as a subscript. So the total demerits Λ_t of a paragraph with μ lines is given by

$$\begin{aligned}\Lambda_\mathrm{t} &= \sum_{\iota=1}^{\mu}\Lambda_\iota = \sum_{\iota=1}^{\mu}\big((\lambda+\beta_\iota)^2+\mathrm{sgn}(\pi_\iota)\pi_\iota^2+\delta_\iota\big)\\ &= \mu\lambda^2+2\lambda\mathrm{B}+\sum_{\iota=1}^{\mu}\big(\beta_\iota^2+\mathrm{sgn}(\pi_\iota)\pi_\iota^2+\delta_\iota\big) \quad (3)\end{aligned}$$

where the sum of all badness values is called B for short, i.e., $\mathrm{B}:=\sum_{\iota=1}^{\mu}\beta_\iota$.

The total demerits of a paragraph sum certain values that are associated with the path through the network of line breaks that TEX has identified as the shortest path according to its cost function. This calculation can be performed for any path in this network so the notation Λ_p for *path demerits* is introduced.

If the value of `\linepenalty` is important it is given as an argument to Λ_t or Λ_p, for example, the left hand side of (3) can be written as $\Lambda_\mathrm{t}(\lambda)$.

The total demerits also have an upper limit. In [4, §833] this limit is coded and it is best to have

$$\Lambda_\mathrm{t} < 2^{30}-1 = 1{,}073{,}741{,}823 \quad (4)$$

otherwise TEX might output overfull lines although line breaks seem to be possible. TEX does not stop working but except for the end of the paragraph it does not create useful feasible breakpoints that are required for TEX's line-breaking decisions [4, §835].

In order to get familiar with the notation a simple lemma is proved. (The symbol '▯' marks the end of a proof or of an example.)

Lemma 1. *If* `\linepenalty` ≥ 0 *and if for a line that has neither penalties nor additional demerits the line demerits are larger than the line demerits of a second line with a penalty* ≥ 0*, additional demerits* ≥ 0*, and a summand* $\epsilon \geq 0$ *then the badness of the first line is larger than the badness of the second.*

Proof: With (1) the lemma claims for two lines with line demerits Λ and Λ' that with $\epsilon \geq 0$

$$\Lambda > \Lambda'+\epsilon \implies \beta > \beta'$$

if there are no penalties and additional demerits in Λ, i.e., $\pi=0$ and $\delta=0$, and if $\pi'\geq 0$ and $\delta'\geq 0$.

$$\begin{aligned}&\Lambda > \Lambda'+\epsilon\\ \iff\quad &(\beta+\lambda)^2 > (\beta'+\lambda)^2+\mathrm{sgn}(\pi')\pi'^2+\delta'+\epsilon\end{aligned}$$

by (1). The sum $\mathrm{sgn}(\pi')\pi'^2+\delta'+\epsilon$ is ≥ 0 as $\pi'\geq 0$, $\delta'\geq 0$, and $\epsilon\geq 0$. It follows that

$$\begin{aligned}&(\beta+\lambda)^2-(\beta'+\lambda)^2 > \pi'^2+\delta'+\epsilon\\ \implies\quad &(\beta-\beta')(\beta+\beta'+2\lambda)>0.\end{aligned}$$

Badness values are ≥ 0 and $\lambda\geq 0$ so that the term $\beta+\beta'+2\lambda$ must be >0. Its product with $\beta-\beta'$ is greater than 0 so that $\beta-\beta'>0$ or $\beta>\beta'$. ▯

Next another well-known property of plain TEX is stated as a lemma.

Lemma 2. *In plain TEX the last line of a paragraph that does not end with a penalty item of value* -10000 *has either badness* 0 *or its glue shrinks.*

Proof: If the last line contains infinite glue the badness is 0 ([3], p. 97).

Otherwise all glue is finite. In plain TEX the `\parfillskip` is defined as `0pt plus 1fil`. Without the `\parfillskip`, which is added by TEX after removing the last glue item in a paragraph ([3, pp. 99–100]), the last line either contains only text or its glue stretches, or shrinks, or has its natural width. In the first two cases the stretchability of `\parfillskip` makes the badness 0. In the last two cases it does not change anything; a line in which the glue has its natural width has badness 0. ▯

3 When is a single line broken by TEX?

Let's start with perhaps the simplest case in which the effect of `\linepenalty` as a factor for TEX's line-breaking decisions can be analyzed: The network of breakpoints allows TEX to typeset either a single line or a pair of lines. Under what conditions does TEX prefer two lines?

Three assumptions are made in this section:

1. The `\linepenalty` is nonnegative, i.e., ≥ 0.
2. The `\parfillskip` is `0pt plus 1fil`.
3. The line width of the second line for the paragraph is wider than the width of the material that is moved from the first to the second line.

Negative values for `\linepenalty` are discussed in Section 5. The reason for the (quite natural) third assumption, which states that a line break produces at most one additional line, is explained in Section 4.

The line demerits of the single line are called Λ_1. The two-line solution is marked with a prime and its line demerits are called Λ'_1 and Λ'_2.

Note that the network must be built from the first pass of the line-breaking algorithm, as a single line is a valid solution. Of course, the single line must shrink its glue, as a line with badness 0 is never a candidate to be typeset in two lines if the user has not entered a `\penalty` command with a negative

value. Without such a penalty a line with badness 0 has the line demerits $\Lambda_1 = \lambda^2$ and a two-line solution must have at least this value for its second line alone: $\Lambda'_2 \geq \lambda^2$. The first line adds the positive value Λ'_1 to the path demerits as no negative penalties are involved, making a two-line solution worse than the one-line result. There is no other case for the glue of the single line as by Lemma 2 the glue cannot stretch.

But let's look at the general case. The path demerits of the single line are computed as

$$\Lambda_\mathrm{p} = \Lambda_1 = (\lambda + \beta_1)^2 \tag{5}$$

because no penalty is added in a first pass; i.e., $\pi_1 = 0$. $\delta_1 = 0$ except if the line is very loose but that cannot happen by Lemma 2; so this summand can be dropped too.

For the two-line solution the calculation is

$$\begin{aligned}\Lambda'_\mathrm{p} &= \Lambda'_1 + \Lambda'_2 \\ &= (\lambda + \beta'_1)^2 + \mathrm{sgn}(\pi'_1){\pi'_1}^2 + \delta'_1 + \lambda^2 + \delta'_2 \end{aligned} \tag{6}$$

because $\beta'_2 = \pi'_2 = 0$ as the second line must have badness 0 because of the assumptions about the line width and the `\parfillskip`. At a break with a hyphen π'_1 is either the value of `\exhyphenpenalty` if the hyphen is part of the text or `\hyphenpenalty` for user entered discretionary hyphens. In both cases the additional demerits of the second line, δ'_2, must contain the value of `\finalhyphendemerits`, called δ_f. And δ'_1 is either 0, or if this line is very loose `\adjdemerits`, named δ_a. In this case δ'_2 contains δ_a too. A break in math or at an explicit `\penalty` does not influence the additional demerits.

TEX will break the text into two lines if and only if $\Lambda_\mathrm{t} = \Lambda'_\mathrm{p} < \Lambda_\mathrm{p}$.

Case 1: No penalty. This means $\pi'_1 = 0$ and δ'_2 does not contain δ_f; thus (6) simplifies to

$$\Lambda'_\mathrm{p} = (\lambda + \beta'_1)^2 + \delta'_1 + \lambda^2 + \delta'_2. \tag{7}$$

A path that generates two lines is preferred by TEX if the right hand side of (7) is smaller than the right hand side of (5):

$$(\lambda + \beta_1)^2 > (\lambda + \beta'_1)^2 + \delta'_1 + \lambda^2 + \delta'_2. \tag{8}$$

To make this inequality true β'_1 must be smaller than β_1 by Lemma 1; the difference is called the "change" χ of badness for the two-line solution, i.e., $\beta_1 - \chi = \beta'_1$ with $\chi > 0$. As the badness β'_1 is greater than or equal to 0 one more inequality is known

$$\beta_1 \geq \chi. \tag{9}$$

All solutions must have a badness of the single line that lies on or above the identity function $g(\chi) = \chi$.

Now $\beta'_1 < 100$ so the first line of the pair is not very loose, thus $\delta'_1 = \delta'_2 = 0$ and inequality (8)

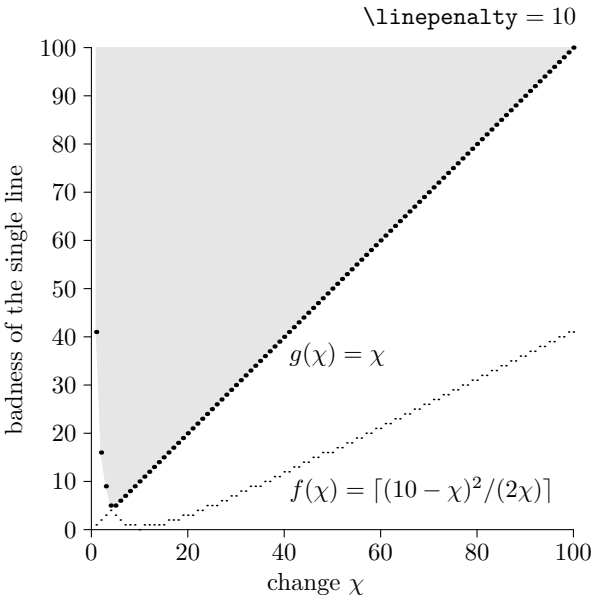

Figure 1: Graphs for functions of Theorem 1

becomes with $\beta'_1 = \beta_1 - \chi$

$$(\lambda + \beta_1)^2 > (\lambda + \beta_1 - \chi)^2 + \lambda^2 \tag{10}$$

$$\iff \qquad 2\beta_1\chi > (\lambda - \chi)^2. \tag{11}$$

As $\chi > 0$, inequality (11) can be written as

$$\beta_1 > \frac{(\lambda - \chi)^2}{2\chi}. \tag{12}$$

With (9) the left hand side of (11) is kept equal or made smaller when β_1 is replaced by χ. If this new inequality holds then (12) holds too.

$$\begin{aligned} & 2\chi^2 > (\lambda - \chi)^2 \\ \iff \quad & \sqrt{2}\chi > \lambda - \chi \qquad \vee \quad \sqrt{2}\chi > \chi - \lambda \\ \iff \quad & \chi > (\sqrt{2} - 1)\lambda \quad \vee \qquad \chi > -(\sqrt{2} + 1)\lambda. \end{aligned}$$

The right-side inequality doesn't say anything new, as $\chi > 0$. If $\lambda = 0$ both inequalities state $\chi > 0$ so that only (12) counts.

This computation proves the following theorem.

Theorem 1. *In plain TEX with* `\linepenalty` ≥ 0 *a text that fits into one line is typeset in two lines containing a line break without penalties if the difference between the badness of the single line and the badness of the first line of the pair is larger than*

$$(\sqrt{2} - 1)\texttt{\textbackslash linepenalty}$$

or this difference, named "change", is larger than zero and the badness of the single line is larger than

$$(\texttt{\textbackslash linepenalty} - \texttt{change})^2/(2 \times \texttt{change}). \qquad \square$$

When plain TEX's settings are used the value 10 can be plugged in for `\linepenalty`. Thus the identity function is used as lower limit for the badness when the change is larger than 4 as $(\sqrt{2} - 1) \times 10 \approx$

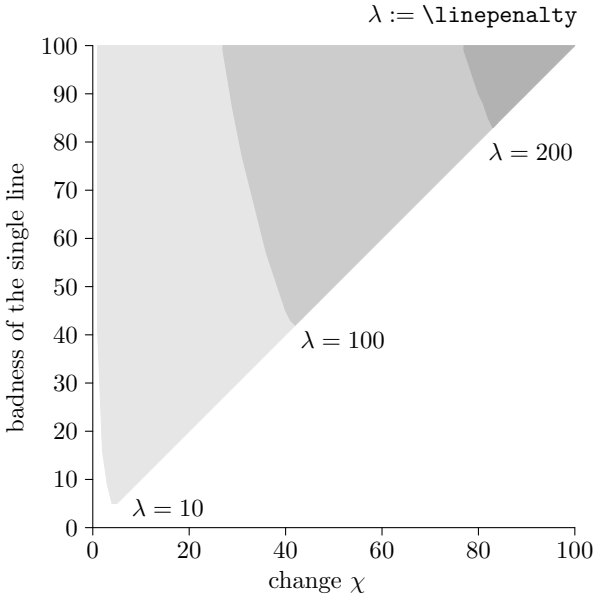

Figure 2: Solution sets for three `\linepenalty` values

4.14. For $(1, 2, 3, 4)$ the badness of the single line must be larger than or equal to $(41, 17, 9, 5)$, respectively; see (12). Figure 1 shows the graphs of two functions for the integer values from 1 to 100. First, the identity is shown as g. The second function f represents in essence the formula of the right hand side of (12). If the badness of the single line lies on or above the thick points for a given χ then two lines are typeset. The gray area forms the *solution set*.

Of course, the `\linepenalty` can be changed. Figure 2 shows the solution sets for three different values of `\linepenalty`: For $\lambda = 10$ all three gray areas count (compare the areas to the solution set shown in Fig. 1), for $\lambda = 100$ the light-colored area is excluded, and for $\lambda = 200$ only the dark area builds the solution set.

Case 2a: Break at hyphen. This important special case of a break with penalties is treated first. The break must be either at an explicit hyphen in the text or at an inserted discretionary break as the network is built from the first pass of TEX's line-breaking algorithm. This means that π'_1 equals either `\exhyphenpenalty` or `\hyphenpenalty` and as explained above there are additional demerits $\delta'_2 =$ `\finalhyphendemerits` $= \delta_f$.

Now (10) and thus (12) get additional constant terms on their right hand sides; (12) becomes:

$$\beta_1 > \frac{(\lambda - \chi)^2 + \operatorname{sgn}(\pi'_1)\pi'^2_1 + \delta_f}{2\chi}.$$

As in plain TEX $\pi'_1 = 50$ and $\delta_f = 5000$, the sum $\operatorname{sgn}(\pi'_1)\pi'^2_1 + \delta_f$ is 7500. A graph for the above inequality similar to Fig. 1 is shown in Fig. 3. The change must be at least 43 to get two lines. With $\beta_1 = 78$ the change must be larger than 75. The comparison of Figs. 1 and 3 shows that in essence the point from which the identity function dominates the other function is moved on this line to a higher value. (The same effect occurs in Fig. 2.)

Case 2b: Break at positive penalty. Lemma 1 is applicable. So starting in (10) with an $\epsilon \geq 0$, which is the sum of the penalty of the first line of the pair and the additional demerits of both lines added to the right hand side, the equivalent of (12) is

$$\beta_1 > \frac{(\lambda - \chi)^2 + \epsilon}{2\chi}.$$

Similarly, starting with inequality (11) and replacing β_1 by χ gives

$$\begin{aligned} & 2\chi^2 > (\lambda - \chi)^2 + \epsilon \\ \iff \quad & (\chi + \lambda)^2 > 2\lambda^2 + \epsilon \\ \iff \quad & \chi > \sqrt{2\lambda^2 + \epsilon} - \lambda \ \vee \ \chi < -\sqrt{2\lambda^2 + \epsilon} - \lambda. \end{aligned}$$

Obviously the second inequality is not relevant.

Thus a generalization of Theorem 1 is proved:

Theorem 2. *Given a text that fits into one line or can be typeset in two lines with a line break that has the value $\epsilon \geq 0$ as the sum of penalties and additional demerits. Let* `\linepenalty` ≥ 0.

The two-line solution is used by plain TEX if the change > 0 is either at least

$$\sqrt{2\texttt{\linepenalty}^2 + \epsilon} - \texttt{\linepenalty}$$

or the badness of the single line is larger than

$$\big((\texttt{\linepenalty} - \texttt{change})^2 + \epsilon\big)/(2 \times \texttt{change}). \ \square$$

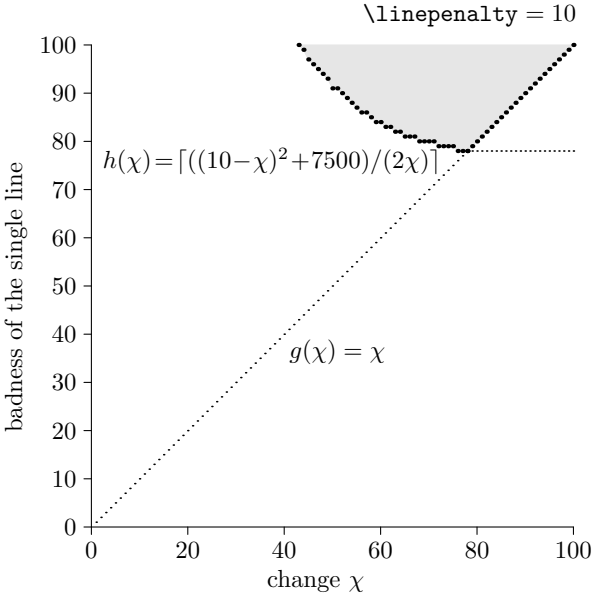

Figure 3: Graphs of functions for a break at hyphen

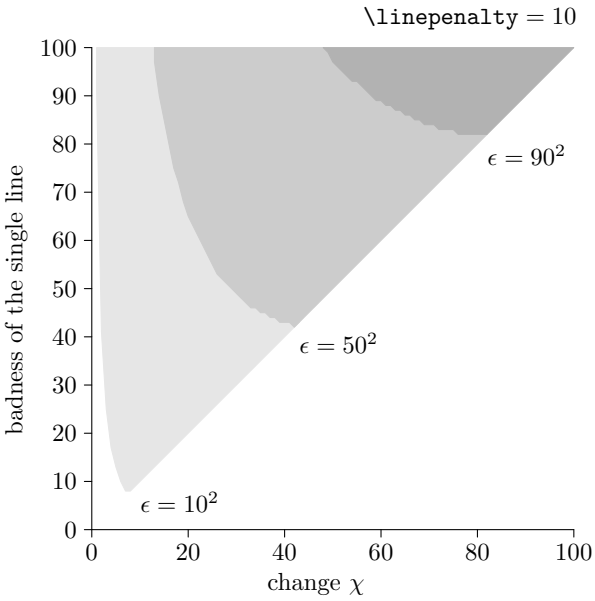

Figure 4: Solution sets for three different penalties

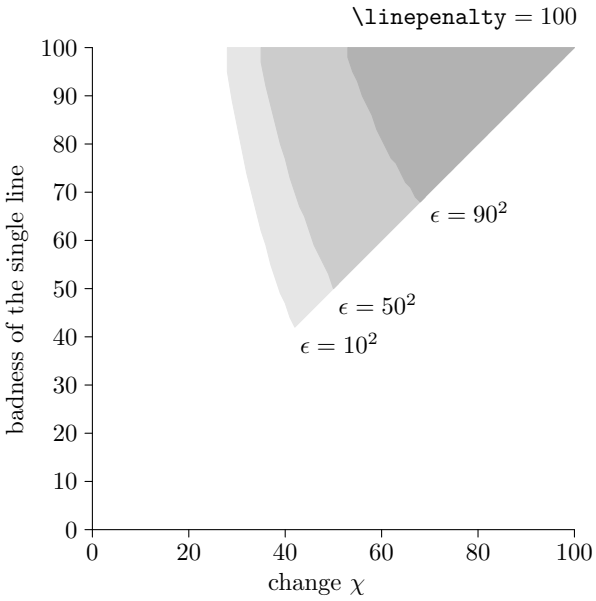

Figure 5: Like Fig. 4 for larger `\linepenalty`

Figure 4 compares the solution sets for three different ϵ similar to Fig. 2.

When the `\linepenalty` is increased to 100 the solution set shrinks as shown in Fig. 2. But Theorem 2 mentions two limits, and the formula $\sqrt{2\lambda^2+\epsilon}-\lambda$ gives for the three penalties 10, 50, and 90, i.e., the three ϵ amounts 10^2, 50^2, and 90^2:

		$\epsilon =$	10^2	50^2	90^2
$\lambda = 10$	$\Longrightarrow$	$\sqrt{200+\epsilon}-10 \approx$	7.3	41.9	81.1
$\lambda = 100$	$\Longrightarrow$	$\sqrt{20000+\epsilon}-100 \approx$	41.7	50	67.6

Thus the values of the limit get larger for $\epsilon = 10^2$ and $\epsilon = 50^2$ when λ is changed to 100 but for $\epsilon = 90^2$ it is smaller! Its solution set is not a subset of the solution set when $\lambda = 10$. Figure 5 shows the solution sets for the three values of ϵ with $\lambda = 100$.

Case 2c: Negative penalties. In this case, the inequality (8) is changed to

$$(\lambda+\beta_1)^2 > (\lambda+\beta_1')^2 + \delta_1' + \lambda^2 - \epsilon \qquad (13)$$

in which $\epsilon > 0$ stands for the sum of penalties of the first line and additional demerits of the second line as in case 2b. That means δ_2' is contained in ϵ and thus not mentioned in (13). $\delta_1' = 0$ except the first line of the two-line solution, named $\mathtt{L}_1'$, is very loose. Then δ_1' is the value of `\adjdemerits` $= \delta_a$. The bracket notation is used to identify this summand: $\delta_a[\mathtt{L}_1'$ very loose$]$. And this means the additional demerits of the second line contains δ_a too. But ϵ is not changed; instead the term δ_a is added twice.

Lemma 1 is not applicable and β_1' can be larger than β_1. For some "change" χ, $-100 \le \chi \le 100$, let $\beta_1 - \chi = \beta_1'$. In other words: Instead of (9), now two limits for the badness of the single line are required:

$$\beta_1 \ge \chi, \qquad \text{if } \chi > 0; \qquad (14a)$$
$$\beta_1 \le 100 + \chi, \qquad \text{if } \chi < 0. \qquad (14b)$$

Inequality (13) becomes

$$(\lambda+\beta_1)^2 > (\lambda+\beta_1-\chi)^2 + 2\delta_a[\mathtt{L}_1' \text{ very loose}] + \lambda^2 - \epsilon$$

or after the usual transformations

$$2\beta_1\chi > (\chi-\lambda)^2 + 2\delta_a[\mathtt{L}_1' \text{ very loose}] - \epsilon.$$

If $\chi = 0$ this inequality states that ϵ must be larger than $\lambda^2 + 2\delta_a[\mathtt{L}_1'$ very loose$]$ in order to typeset two lines.

If $\chi > 0$ then $\beta_1' \ne 100$, i.e., $\mathtt{L}_1'$ cannot be very loose and

$$\beta_1 > \frac{(\chi-\lambda)^2 - \epsilon}{2\chi}. \qquad (15)$$

As before χ is used to replace β_1 with (14a) to get

$$2\chi^2 > \chi^2 - 2\lambda\chi + \lambda^2 - \epsilon$$
$$\Longleftrightarrow \qquad (\chi+\lambda)^2 > 2\lambda^2 - \epsilon$$
$$\Longleftrightarrow \chi > \sqrt{2\lambda^2-\epsilon} - \lambda \ \vee \ \chi < -\sqrt{2\lambda^2-\epsilon} - \lambda.$$

Only the first inequality states something new: If $\epsilon \ge 2\lambda^2$ two lines are typeset. Otherwise $\epsilon < 2\lambda^2$ and either $\chi > \sqrt{2\lambda^2-\epsilon} - \lambda$ or the badness of the first line fulfilling (15) are needed to get two lines.

If $\chi < 0$ the first line of the two-line solution might be very loose and χ must obey (14b).

First, the inequality (15) changes to

$$\beta_1 < \frac{(\chi-\lambda)^2 + 2\delta_a[\mathtt{L}_1' \text{ very loose}] - \epsilon}{2\chi}. \qquad (16)$$

Starting from

$$2\beta_1\chi > (\chi-\lambda)^2 + 2\delta_a[\mathtt{L}_1' \text{ very loose}] - \epsilon$$

as above, now (14b) must be used to replace β_1:

$$2(100+\chi)\chi > (\chi-\lambda)^2 + 2\delta_a[\mathrm{L}_1' \text{ very loose}] - \epsilon.$$

With the usual transformations this leads to the relevant solution

$$\chi > \sqrt{(100+\lambda)^2+\lambda^2+2\delta_a[\mathrm{L}_1' \text{ very loose}]-\epsilon} - 100 - \lambda.$$

Thus a somewhat complex third theorem is proved:

Theorem 3. *Given a text that fits into one line or can be typeset in two lines with a line break that has the value $-\epsilon < 0$ as the sum of penalties and additional demerits except for* `\adjdemerits` *if the first line is very loose. Let* `\linepenalty` ≥ 0.

If the change is 0 *then there are two cases: If the first line of the pair is very loose ϵ must be larger than* `\linepenalty`$^2 + 2$`\adjdemerits`*; otherwise $\epsilon >$* `\linepenalty`^{2} *is sufficient.*

If the change is > 0 then $\epsilon \geq 2$`\linepenalty`^{2} *typeset two lines; otherwise if $\epsilon < 2$*`\linepenalty`^{2} *then either the change must be larger than*

$$\sqrt{2\texttt{\textbackslash linepenalty}^2-\epsilon} - \texttt{\textbackslash linepenalty}$$

or the badness of the single line is larger than

$$\frac{(\texttt{change} - \texttt{\textbackslash linepenalty})^2 - \epsilon}{2 \times \texttt{change}} \qquad (*)$$

to output two lines.

If the change is smaller than 0 *but the first line is not very loose and $\epsilon \geq (100+\lambda)^2+\lambda^2$ then two lines are created. Otherwise if ϵ is smaller then either the change must be larger than*

$$\sqrt{(100+\texttt{\textbackslash linepenalty})^2+\texttt{\textbackslash linepenalty}^2-\epsilon} - 100 - \texttt{\textbackslash linepenalty}$$

or the badness of the single line must be smaller than $(*)$ to output two lines.

If the first line of the pair is very loose then two lines are typeset if either the change is smaller than

$$\sqrt{(100+\texttt{\textbackslash linepenalty})^2+\texttt{\textbackslash linepenalty}^2 + 2\texttt{\textbackslash adjdemerits}-\epsilon} - 100 - \texttt{\textbackslash linepenalty}$$

and $(*)$ + `\adjdemerits`/`change` *is larger than the badness of the single line or ϵ is larger than*

$$(100+\texttt{\textbackslash linepenalty})^2+\texttt{\textbackslash linepenalty}^2 + 2\texttt{\textbackslash adjdemerits}. \quad \square$$

Figure 6 shows in the style of previous figures three instances of negative penalties. All gray areas represent the solution set if $\epsilon = 159^2$. The dots show the limit when the first line of the pair is very loose. If the dots and the lightest gray area are excluded the diagram shows the solution set for $\epsilon = 68^2$. It

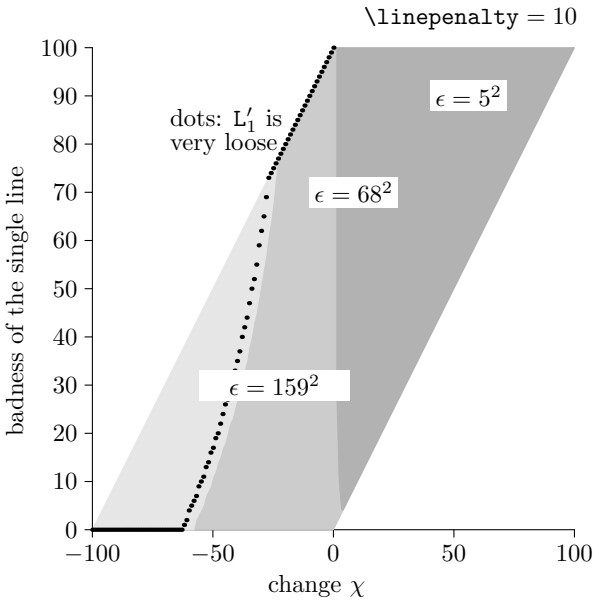

Figure 6: Solution sets for three negative values

doesn't capture all negative change but the smallest positive values build the identity function. If $\epsilon = 5^2$ only the darkest area counts and even for positive change the identity function is not achieved for small values.

Setting `\linepenalty` $= 100$ makes all areas smaller, the dots disappear, and for $\epsilon = 159^2$ the left edge of the solution set drops from $(-43, 57)$ to $(-59, 0)$.

4 A few consequences of the theorems

The developed theory helps to understand certain cases for plain TEX when a single line can be broken.

Theorem 1 shows how the two-line solution can be made more likely when no penalties are involved: Reduce the value of `\linepenalty`! The assignment 2 requires a change that must be larger than $2(\sqrt{2}-1) < 1$ for all badness values, i.e., a change of 1 or more typesets the two lines.

Theorem 1 also states that with a large value of `\linepenalty` TEX will typeset a single line, for example, with a value 242 the line break is impossible as the change must be larger than $242(\sqrt{2}-1) > 100$.

Theorem 2 makes, among other things, statements about penalties. It proves that a penalty of 110 forces the single line as $\big((10-100)^2+110^2\big)/(2\times 100) > 100$.

Theorem 3 implies that a `\penalty` of -180 typesets always two lines even if the first line of this pair is very loose and `\adjdemerits` are involved.

Larger `\linepenalty` might break a single line. One interesting consequence of Theorem 2 is that a

larger \linepenalty in combination with a positive \penalty breaks a line that would be kept as a single line if the default value of \linepenalty is used. See the discussion after Theorem 2 comparing Figs. 4 and 5.

Example 1: Description
A single line is broken when \linepenalty is increased.

TEX input

```
\toks0={\noindent It's a surprise, but it's true.
   See for yourself now. So~it\penalty95\ is.}
\linepenalty=10  \the\toks0\par
\linepenalty=100 \the\toks0\par
```

TEX output
It's a surprise, but it's true. See for yourself now. So it is. It's a surprise, but it's true. See for yourself now. So it is. ∎

The single line has badness 86 and without the "is." the badness drops to 0, that is, the first line of the two-line solution produces a change of 86. Using the formula of Theorem 2, once the values 10 and 95^2 and 100 and 95^2 are used for \linepenalty and for ϵ, respectively, the results are that the change must be larger than 86 in the first case and larger than 70 in the second to create two lines. Thus, in the first case the break is avoided and in the second it is made.

Instead of an explicit penalty a hyphen can be the reason for a line break:

Example 1 continued: TEX input

```
\toks0={Bob, tell us, what made you
        want to look up {\sl run-in\/}?}
\linepenalty=10  \the\toks0\par
\linepenalty=100 \the\toks0\par
```

TEX output
Bob, tell us, what made you want to look up *run-in*? Bob, tell us, what made you want to look up *run-in*? ∎

Three lines with one line break. The theory was developed with the starting point that a line break generates two lines; see assumption 3 in Section 3. But TEX is very flexible and a user can construct situations in which a line break generates *two additional* lines.

Example 2: Description
An unusual \parshape is presented that otherwise is not considered in this article.

TEX input

```
\def\weirdparshape{\setbox0=\hbox{\ninerm is}
   \parshape 3 0pt \hsize 0pt \wd0 0pt \hsize
   Do! not! do! it! Never! No!
   This \cs{parshape} is bad.}
\linepenalty=242 \weirdparshape\par
\linepenalty=10  \weirdparshape\par
```

TEX output
Do! not! do! it! Never! No! This \parshape is bad. Do! not! do! it! Never! No! This \parshape is bad. ∎

Of course, the theory could be extended, but it does not seem worth the effort. Such settings of \parshape are never applied to normal text. The author wants to generate a certain effect and controls the situation.

Longer paragraphs. The theoretical results do not apply without change to paragraphs with more than one line because the penultimate line in a long paragraph might be changed too when the last line is broken, i.e., $(\mathtt{L}_{\mu-1}, \mathtt{L}_\mu) \to (\mathtt{L}'_{\mu-1}, \mathtt{L}'_\mu, \mathtt{L}'_{\mu+1})$ with $\mathtt{L}_{\mu-1} \neq \mathtt{L}'_{\mu-1}$. And even if it stays unchanged, i.e., $\mathtt{L}_{\mu-1} = \mathtt{L}'_{\mu-1}$, the line characteristic might influence the next line through additional demerits in different ways. Finally, the paragraph might be broken in the second pass of TEX's line-breaking algorithm, thus the badness of $\mathtt{L}'_\mu$ might be larger than the badness of $\mathtt{L}_\mu$ even if the change is positive.

OK, enough warning notices: There are nevertheless cases in which the theory is applicable to longer paragraphs.

Example 3: Description
Typeset a short text twice with plain TEX. First with the default settings, next with \linepenalty = 2.

TEX output
When you start to count where do you start? With zero or with one? Hmm, I start at 1! A CS nerd uses a 0, or? When you start to count where do you start? With zero or with one? Hmm, I start at 1! A CS nerd uses a 0, or? ∎

Later in Section 5 it is shown that the value 4 for \linepenalty is sufficient. It turns out that the value −2 works in this case too; see Section 6.

Next the technique with the penalty of 110 is used. The example also demonstrates that a large \linepenalty does not break the last line if the theory is applicable. Here the minimal required value for \linepenalty is used.

Example 4: Description
Typeset a short text thrice with plain TEX. First with plain TEX's default settings, second with a \penalty110 inserted between the last two words and a third time without this penalty but with \linepenalty = 199.

TEX definitions

```
\toks0{\noindent This text can be typeset, yes,
either in two or in three lines and the theory
of this section applies to the}
\toks1={\the\toks0{} last line.}
\toks2={\the\toks0{} last\penalty110\ line.}
```

TEX input

```
\linepenalty=10 \the\toks1\par \the\toks2\par
\linepenalty=199 \the\toks1\par
```

TEX output

This text can be typeset, yes, either in two or in three lines and the theory of this section applies to the last line.
This text can be typeset, yes, either in two or in three lines and the theory of this section applies to the last line.
This text can be typeset, yes, either in two or in three lines and the theory of this section applies to the last line.

As expected the `\penalty110` prevents the line break. A tie would do the job too but if the text later grows, a break at the `\penalty` is still possible.

On the other hand if the text is typeset twice in one paragraph the theory is not applicable; with a line break in the last line the word "three" is moved from the penultimate line of the four-line paragraph to the penultimate line of the five-line paragraph.

Example 5: Description

Typeset the paragraph of example 4 two times as one paragraph: once with plain TEX's defaults and once with `\linepenalty` = 385.

TEX input

```
\linepenalty=10  \the\toks1{} \the\toks1\par
\linepenalty=385 \the\toks1{} \the\toks1\par
```

TEX output

This text can be typeset, yes, either in two or in three lines and the theory of this section applies to the last line. This text can be typeset, yes, either in two or in three lines and the theory of this section applies to the last line.
This text can be typeset, yes, either in two or in three lines and the theory of this section applies to the last line.
This text can be typeset, yes, either in two or in three lines and the theory of this section applies to the last line.

In order to bring this paragraph to four lines `\linepenalty` must be set to 385.

5 Changing the value of `\linepenalty`

The parameter `\linepenalty` can be changed by the user. In this section an analysis is made when a different `\linepenalty` results in different line breaks and what the trade-offs are.

Example 6: Description

Typeset a paragraph several times with different values for `\linepenalty`. Start with TEX's default settings.

TEX definitions

```
\linepenalty=10
```

TEX output

The line-breaking algorithm of TEX selects a shortest path in a network of feasible breakpoints using a cost function that calculates *demerits*. For every line four values are used to compute the demerits for this line and then the sum of all line demerits stands for the total demerits of a paragraph.

The line-breaking decisions of TEX are listed in the log file if `\tracingparagraphs` is set to 1. This output helps to explain the effect on the line breaks when `\linepenalty` is changed; therefore the trace data is shown. See *The TEXbook* [3], pp. 98–99, or [8], Section 3, for a description of this data.

Example 6 continued: `\tracingparagraphs`' data

```
1. @firstpass
2. @secondpass
3. []\ninerm The line-breaking al-go-rithm of
   T[]X se-lects a short-
4. @\discretionary via @@0 b=5 p=50 d=2725
5. @@1: line 1.2- t=2725 -> @@0
6. est path in a net-work of fea-si-ble break-
   points us-ing a
7. @ via @@1 b=82 p=0 d=8464
8. @@2: line 2.1 t=11189 -> @@1
9. cost
10. @ via @@1 b=20 p=0 d=900
11. @@3: line 2.3 t=3625 -> @@1
12. func-tion that cal-cu-lates \ninesl
    de-mer-its\ninerm . For ev-ery line
13. @ via @@2 b=48 p=0 d=3364
14. @@4: line 3.1 t=14553 -> @@2
15. four
16. @ via @@3 b=45 p=0 d=13025
17. @@5: line 3.1 t=16650 -> @@3
18. val-
19. @\discretionary via @@3 b=83 p=50 d=11149
20. @@6: line 3.3- t=14774 -> @@3
21. ues are used to com-pute the de-mer-its for
    this
22. @ via @@4 b=39 p=0 d=2401
23. @@7: line 4.1 t=16954 -> @@4
24. line
25. @ via @@4 b=36 p=0 d=12116
26. @ via @@5 b=63 p=0 d=5329
27. @@8: line 4.1 t=21979 -> @@5
28. @@9: line 4.3 t=26669 -> @@4
29. and
30. @ via @@5 b=20 p=0 d=10900
31. @ via @@6 b=5 p=0 d=225
32. @@10: line 4.2 t=14999 -> @@6
33. then the sum of all line de-mer-its stands
    for
34. @ via @@7 b=84 p=0 d=8836
35. @@11: line 5.1 t=25790 -> @@7
36. the
37. @ via @@7 b=0 p=0 d=100
38. @ via @@8 b=114 p=0 d=15376
39. @ via @@9 b=114 p=0 d=25376
40. @@12: line 5.2 t=17054 -> @@7
41. to-
42. @\discretionary via @@8 b=0 p=50 d=2600
43. @\discretionary via @@9 b=0 p=50 d=2600
```

```
@@13: line 5.2- t=24579 -> @@8
tal
@ via @@8 b=15 p=0 d=10625
@ via @@9 b=15 p=0 d=625
@ via @@10 b=46 p=0 d=3136
@@14: line 5.1 t=18135 -> @@10
@@15: line 5.3 t=27294 -> @@9
de-
@\discretionary via @@10 b=3 p=50 d=2669
@@16: line 5.2- t=17668 -> @@10
mer-its of a para-graph.
@\par via @@11 b=0 p=-10000 d=100
@\par via @@12 b=0 p=-10000 d=100
@\par via @@13 b=0 p=-10000 d=5100
@\par via @@14 b=0 p=-10000 d=100
@\par via @@15 b=0 p=-10000 d=100
@\par via @@16 b=0 p=-10000 d=5100
@@17: line 6.2- t=17154 -> @@12
```

Here are a few reasons why this paragraph is a good candidate to see the effect of different values for `\linepenalty`.

Reason 1: The first line breaks at a hyphen so the paragraph needs a second pass; see lines 1–2 of the listing. Thus penalties might occur, the additional demerits are not limited to `\adjdemerits`, and very loose lines are possible.

Reason 2: There are many possible paths in the network; see lines 55–60. (Of course, this is normal for most longer paragraphs.) Thus there are other ways to typeset the text.

Reason 3: Some lines have a rather high badness, but it is possible by adding a word from the neighboring line to lower the badness dramatically; for example, see lines 6–10. The shortest path contains some lines that have one of those large badness values; see lines 4, 7, 13, 22, 37.

Reason 4: Some of the possible line breaks for a penultimate line makes this line end with a hyphen (lines 51–53 and 60 as well as lines 41–44 and 57). So `\finalhyphendemerits` are available as additional demerits.

Reason 5: On the other hand, some possible lines avoiding the hyphen at the end of the penultimate line are very loose; see lines 38–40. Thus very loose lines are indeed available, and not just a possibility as stated in reason 1.

Reasons 1–4 are useful to see an effect for higher positive values of `\linepenalty`, the last one to see an effect if the value is negative.

Table 1 summarizes the paths identified in lines 55–60 of the trace data. The table shows in the first two columns the information of the `@@`-lines: the sequence number and the fitness class abbreviated to the first letter of very loose, loose, decent, or tight. Then six columns for the possible paths are presented (lines 55–60); the heading gives the sequence number after the "`via @@`"; the subscript 0 is explained later. The table entries are the badness values. A subscript signals that a penalty occurs at the break, a superscript of 'f' or 'a' that `\finalhyphendemerits` or `\adjdemerits`, respectively, are applied. Line 61 of the listing reports that the line breaks follow the path of the column labeled 12_0. The column head contains an asterisk to indicate this selection by TEX.

Table 1: Badness, penalties, and additional demerits of the line breaks for the six paths of the trace listing

		`\par via @@` (`*` is typeset)					
`@@`	Class	11_0	$*12_0$	13_0	14_0	15_0	16_0
1	d	5_{50}	5_{50}	5_{50}	5_{50}	5_{50}	5_{50}
2	l	82	82			82	
3	t			20	20		20
4	l	48	48			48	
5	l			45^{a}			
6	t				83_{50}		83_{50}
7	l	39	39				
8	l			63			
9	t					36^{a}	
10	d				5		5
11	l	84					
12	d		0				
13	d			0_{50}			
14	l				46		
15	t					15	
16	d						3_{50}
17	d	0	0	0^{f}	0	0	0^{f}
	$\mu =$	6	6	6	6	6	6
	B =	258	174	133	159	186	116
$\Lambda_p(10) =$		25890	17154	29679	18235	27394	22768

The last three rows state the number of lines, μ, the sum of the badness values of the path, B, and the path demerits Λ_p. These values are not found directly in the trace data. They have been computed from the information in the columns.

The theory. An increase of λ by $\kappa > 0$ changes the first summand of the formula (1) for the line demerits

$$(\lambda + \kappa + \beta)^2 = (\lambda + \beta)^2 + 2\beta\kappa + 2\lambda\kappa + \kappa^2.$$

The two summands $2\lambda\kappa$ and κ^2 form a "constant" that is added to every line and therefore they do not change the line-breaking decisions by TEX — as long as the limits (2) and (4) are obeyed. The third summand $2\beta\kappa$ increases the influence of the badness β. That means, the penalties and the additional demerits in (1) are less important: If κ is large enough TEX selects a path for which more penalties or additional demerits are charged if only the badness values can be made smaller.

In fact, the increment that is necessary to go from one path to another can be calculated with (1) and (3). The path demerits become

$$\begin{aligned}\Lambda_{\mathrm{p}}(\lambda+\kappa) &= \sum_{\iota=1}^{\mu}\bigl((\lambda+\kappa+\beta_\iota)^2+\mathrm{sgn}(\pi_\iota)\pi_\iota^2+\delta_\iota\bigr)\\ &= \mu(\lambda+\kappa)^2+2\kappa\mathrm{B}+\Lambda_{\mathrm{p}}(\lambda)-\mu\lambda^2.\end{aligned}$$

The task is to determine $\kappa > 0$ to change the total line demerits to a path with a lower sum of badness — this path gets all subscripted variables and their sums primed. Therefore starting with $\mathrm{B}' < \mathrm{B}$ and $\Lambda_{\mathrm{t}}(\lambda) = \Lambda_{\mathrm{p}}(\lambda) < \Lambda'_{\mathrm{p}}(\lambda)$ find $\kappa > 0$ such that

$$\Lambda_{\mathrm{p}}(\lambda+\kappa) > \Lambda'_{\mathrm{p}}(\lambda+\kappa) = \Lambda_{\mathrm{t}}(\lambda+\kappa).$$

In the longer form the inequality is

$$\mu(\lambda+\kappa)^2+2\kappa\mathrm{B}+\Lambda_{\mathrm{p}}(\lambda)-\mu\lambda^2 > \mu'(\lambda+\kappa)^2+2\kappa\mathrm{B}'+\Lambda'_{\mathrm{p}}(\lambda)-\mu'\lambda^2.$$

A few simple transformations when $\mu = \mu'$ give

$$2\kappa(\mathrm{B}-\mathrm{B}') > \Lambda'_{\mathrm{p}}(\lambda)-\Lambda_{\mathrm{p}}(\lambda)$$

or as $\mathrm{B} > \mathrm{B}'$

$$\kappa > \frac{\Lambda'_{\mathrm{p}}(\lambda)-\Lambda_{\mathrm{p}}(\lambda)}{2(\mathrm{B}-\mathrm{B}')}. \tag{17}$$

Its application. Table 1 shows that there are three paths with a lower sum of badness value than column $\mathtt{12}_0$: $\mathtt{14}_0$ with sum 159, $\mathtt{13}_0$ with sum 133, and $\mathtt{16}_0$ with sum 116. Inequality (17) states the following conditions for κ:

Path (Column)	$\mathtt{13}_0$	$\mathtt{14}_0$	$\mathtt{16}_0$
$\kappa >$	152	36	48

(Note, only integer parts of the numbers are shown.)

So $\kappa = 37$ (or $\lambda = 47$) typesets the path of column $\mathtt{14}_0$. The path of column $\mathtt{13}_0$ cannot be reached: $\mathtt{16}_0$ has a lower sum of badness and needs a lower κ.

Example 6 continued: TEX definitions

```
\linepenalty=47
```

TEX output

The line-breaking algorithm of TEX selects a shortest path in a network of feasible breakpoints using a cost function that calculates *demerits*. For every line four values are used to compute the demerits for this line and then the sum of all line demerits stands for the total demerits of a paragraph. ∎

Of course, the typeset result has one more hyphenated line. Low badness values have been traded in for more penalties. A value $\kappa > 48$ selects the path of column $\mathtt{16}_0$ but it might not be the value 49. The formula does not know that there is a column in between; so 49 still creates the path of column $\mathtt{14}_0$. Using (17) the calculation of κ to go from column $\mathtt{14}_0$ to column $\mathtt{16}_0$ gives $\kappa = 53$ or $\lambda = 63$.

Example 6 continued: TEX definitions

```
\linepenalty=63
```

TEX output

The line-breaking algorithm of TEX selects a shortest path in a network of feasible breakpoints using a cost function that calculates *demerits*. For every line four values are used to compute the demerits for this line and then the sum of all line demerits stands for the total demerits of a paragraph. ∎

All paths with a smaller sum of badness have been used. But these are not all possible paths as TEX ignores a path that cannot become the shortest. Except `\linepenalty` $\neq 10$ might change TEX's viewpoint but unfortunately the path is not shown explicitly in the available trace. In total there are six more paths hidden in the data; Table 1′ lists them. The paths are still named by the `par` information and now the subscript identifies the variant. The notations `(10)` and `(t)` mean that the corresponding `@`-line does not have its own `@@`-line in the trace and `@@10` follows next. With this data all values of κ for the transitions to a path with a smaller sum of badness can be computed if the paths are sorted by their sum of badness values B:

	$\mathtt{16}_1$	$\mathtt{16}_0$	$\mathtt{13}_0$	$\mathtt{14}_1$	$\mathtt{15}_1$	$\mathtt{14}_0$	$\mathtt{13}_1$	$\mathtt{12}_0$
$\mathtt{16}_1$	0							
$\mathtt{16}_0$	272	0						
$\mathtt{13}_0$	70	−204	0					
$\mathtt{14}_1$	52	−201	−185	0				
$\mathtt{15}_1$	23	−156	−101	−80	0			
$\mathtt{14}_0$	129	52	220	272	657	0		
$\mathtt{13}_1$	6	−106	−62	−52	−37	−673	0	
$\mathtt{12}_0$	112	48	152	179	299	36	2869	0

As noted above 36 is the smallest number in the last row, selecting path $\mathtt{14}_0$. In the row for $\mathtt{14}_0$ 52 is the smallest number selecting $\mathtt{16}_0$ and in its row 272 is selecting $\mathtt{16}_1$. So only one more path can be shown for $\kappa > 272$, i.e., κ must be 273 and $\lambda = \kappa+10 = 283$.

Example 6 continued: TEX definitions

```
\linepenalty=283
```

TEX output

The line-breaking algorithm of TEX selects a shortest path in a network of feasible breakpoints using a cost function that calculates *demerits*. For every line four values are used to compute the demerits for this line and then the sum of all line demerits stands for the total demerits of a paragraph. ∎

Negative values. A negative amount for the integer `\linepenalty` does not act directly as a bonus if a line is created, as the sum with the badness is squared in equation (1). But a negative value reverts the meaning of badness! For example, a value

Table 1′: Badness, penalties, and additional demerits of the line breaks for not-shown paths of the trace listing

		variant of `\par via @@`					
`@@`	Class	$\texttt{12}_2$	$\texttt{12}_1$	$\texttt{13}_1$	$\texttt{14}_1$	$\texttt{15}_1$	$\texttt{16}_1$
1	d	5_{50}	5_{50}	5_{50}	5_{50}	5_{50}	5_{50}
2	l	82		82			
3	t		20		20	20	20
4	l	48		48			
5	l		45^{a}		45^{a}	45^{a}	45^{a}
8	l		63			63	
9	t	36^{a}		36^{a}			
(10)	(t)				20^{a}		20^{a}
(12)	(v)	114^{a}	114				
13	d			0_{50}			
14	l				46		
15	t					15^{a}	
16	d						3_{50}
17	d	0^{a}	0	0^{f}	0	0	0^{f}
	$\mu =$	6	6	6	6	6	6
	B =	285	247	171	136	148	93
$\Lambda_p(10) =$		62145	37455	34369	30786	32704	35319

of −110 for `\linepenalty` assigns lines with badness 0 the same demerits as lines with badness 100 get with plain TEX's default settings. And a line with badness 100 gets the value that previously a line with badness 0 received. TEX creates lines that have large badness values if possible! Higher negative values retain this effect, so they act differently from large positive values.

Tables 1 and 1′ show that there are paths that have a larger sum of badness than the path of column $\texttt{12}_0$. Inequality (17) is changed as in this case $\mathrm{B} < \mathrm{B}'$. Thus division by $2(\mathrm{B} - \mathrm{B}') < 0$ inverts the relation:

$$\kappa < \frac{\Lambda'_p(\lambda) - \Lambda_p(\lambda)}{2(\mathrm{B} - \mathrm{B}')}. \qquad (18)$$

As in the case of smaller sum of badness values a diagonal matrix with values that are larger than B of $\texttt{12}_0$ can be built.

	$\texttt{12}_0$	$\texttt{15}_0$	$\texttt{12}_1$	$\texttt{11}_0$	$\texttt{12}_2$
$\texttt{12}_0$	0	−426	−139	−52	−202
$\texttt{15}_0$		0	−82	11	−175
$\texttt{12}_1$			0	526	−324
$\texttt{11}_0$				0	−671
$\texttt{12}_2$					0

So this time $\kappa = -53$, i.e., $\lambda = -43$, selects column $\texttt{11}_0$.

Example 6 continued: TEX definitions

```
\linepenalty=-43
```

TEX output

The line-breaking algorithm of TEX selects a shortest path in a network of feasible breakpoints using a cost function that calculates *demerits*. For every line four values are used to compute the demerits for this line and then the sum of all line demerits stands for the total demerits of a paragraph. ∎

The matrix states that from row $\texttt{11}_0$ the value $\kappa = -672$ moves on to path $\texttt{12}_2$.

Example 6 continued: TEX definitions

```
\linepenalty=-662
```

TEX output

The line-breaking algorithm of TEX selects a shortest path in a network of feasible breakpoints using a cost function that calculates *demerits*. For every line four values are used to compute the demerits for this line and then the sum of all line demerits stands for the total demerits of a paragraph. ∎

The limit. The `\linepenalty` has a limit as stated in (2). The setting of 10000 is larger than this limit and the badness values are completely ignored — they do not even have a "little influence" [1, p. 171]. (The effects that are shown in [1] can neither be reproduced with the font `cmr10` nor are they explained by the developed theory.) As badness plays no rôle anymore, TEX tries to avoid hyphens and visually incompatible lines as they add to the line demerits; see (1). In the current example the same line breaks as with `\linepenalty` = 10 are used.

But other paragraphs, with a `\linepenalty` value above the limit (2), switch to a path that otherwise can be reached only by a negative penalty.

Table 2: Badness, penalties, and additional demerits of the line breaks for the two paths of example 7

		`\par via @@` (`*` is typeset)	
`@@`	Class	`*6`	`7`
1	v		
2	d	0_{50}	
3	t		68
4	d	0	
5	d		5
6	l	32	
7	d		0
8	d	0	0
	$\mu =$	4	4
	B =	32	73
$\Lambda_p(10) =$		4564	6509

Example 7: Description

Typeset a paragraph three times in a forced second pass, i.e., `\pretolerance` = −1: first with `\linepenalty` = 10, second with `\linepenalty` = −14, and third with `\linepenalty` = 10000.

TEX output

Hi! TEX! Tell me: How is the following long word broken 'pneumonoultramicroscopicsilicovolcanoconiosis'? I am sure that you are an expert in hyphenation, right TEX? Or shall I ask Siri?

Hi! TEX! Tell me: How is the following long word broken 'pneumonoultramicroscopicsilicovolcanoconiosis'? I am sure that you are an expert in hyphenation, right TEX? Or shall I ask Siri?
Hi! TEX! Tell me: How is the following long word broken 'pneumonoultramicroscopicsilicovolcanoconiosis'? I am sure that you are an expert in hyphenation, right TEX? Or shall I ask Siri? ∎

Table 2 shows: $\kappa < -23.7 \approx (6509-4564)/-82$ by (18), that is $\lambda = -14$, selects the path in column 7. As the path in column 6 contains a hyphen a `\linepenalty` of 10000 selects the path in column 7, which has no penalties or additional demerits.

Paths with fewer lines. In example 5, the parameter `\linepenalty` must be set to a large value in order to reduce the number of lines that are typeset from five to four. This is not covered by (17) as now $\mu = \mu' + 1$.

To determine the κ that selects a path with fewer lines the initial inequality must distinguish between μ and $\mu' = \mu - 1$:

$$\mu(\lambda+\kappa)^2 + 2\kappa\mathrm{B} + \Lambda_\mathrm{p}(\lambda) - \mu\lambda^2 > (\mu-1)(\lambda+\kappa)^2 + 2\kappa\mathrm{B}' + \Lambda'_\mathrm{p}(\lambda) - (\mu-1)\lambda^2.$$

The difference between μ and μ' adds the summand $(\lambda+\kappa)^2 - \lambda^2$ to the left hand side and this allows that the sum of badness can be larger for the shorter paragraph.

A simple rearrangement of the terms gives

$$\kappa^2 + 2\kappa(\lambda + \mathrm{B} - \mathrm{B}') > \Lambda'_\mathrm{p}(\lambda) - \Lambda_\mathrm{p}(\lambda). \qquad (19)$$

Addition of $(\lambda + \mathrm{B} - \mathrm{B}')^2$ to both sides gives a quadratic term on the left

$$(\kappa + \lambda + \mathrm{B} - \mathrm{B}')^2 > (\lambda + \mathrm{B} - \mathrm{B}')^2 + \Lambda'_\mathrm{p}(\lambda) - \Lambda_\mathrm{p}(\lambda)$$

and as the right hand side is positive, the square root can be taken. Therefore the following relevant inequality is found

$$\kappa > \mathrm{B}' - \mathrm{B} - \lambda + \sqrt{(\lambda + \mathrm{B} - \mathrm{B}')^2 + \Lambda'_\mathrm{p}(\lambda) - \Lambda_\mathrm{p}(\lambda)}\,. \qquad (20)$$

Table 3 shows the data of the corresponding trace listing for the text of example 5. The notation `(10)` is explained above; see also "No information in the trace data" in [8], p. 370ff. The path of column 8 is typeset and it has the lowest value for the sum of badness B. Using this data inequality (20) gives $\kappa > 374.8\ldots$ for column 6. The required `\linepenalty` must be set to 385 — as mentioned after example 5.

Paths with more lines. In example 3 the parameter `\linepenalty` was set to 2 to enlarge the number of typeset lines. So this case should be analyzed too.

Table 3: Badness, penalties, and additional demerits of the line breaks for the four paths of example 5

		`\par via @@` (`*` is typeset)			
`@@`	Class	`6`	`7`	`*8`	`9`
`1`	d	2	2	2	2
`2`	l		19	19	
`3`	t	96			96
`4`	l		87		
`5`	d			5	
`6`	d	2			2
`7`	l		13		
`8`	d			1	
`9`	l				19_{110}
`(10)`	(t)	96			
`10`	d		0	0	0
	$\mu =$	4	5	5	5
	B =	196	121	27	119
$\Lambda_\mathrm{p}(10) =$		22760	11023	1431	24565

This time $\kappa > 0$ is subtracted from λ:

$$\Lambda_\mathrm{p}(\lambda-\kappa) = \sum_{\iota=1}^{\mu}\bigl((\lambda-\kappa+\beta_\iota)^2 + \mathrm{sgn}(\pi_\iota)\pi_\iota^2 + \delta_\iota\bigr) = \mu(\lambda-\kappa)^2 - 2\kappa\mathrm{B} + \Lambda_\mathrm{p}(\lambda) - \mu\lambda^2.$$

Thus the inequality for the path demerits becomes

$$\mu(\lambda-\kappa)^2 - 2\kappa\mathrm{B} + \Lambda_\mathrm{p}(\lambda) - \mu\lambda^2 < \mu'(\lambda-\kappa)^2 - 2\kappa\mathrm{B}' + \Lambda'_\mathrm{p}(\lambda) - \mu'\lambda^2$$

and with $\mu' = \mu + 1$ this is

$$\Lambda_\mathrm{p}(\lambda) - \Lambda'_\mathrm{p}(\lambda) < \kappa^2 - 2\kappa(\mathrm{B} - \mathrm{B}' - \lambda).$$

This time add the term $(\lambda + \mathrm{B}' - \mathrm{B})^2$ to both sides and take square roots; then the useful result is

$$\kappa > \lambda + \mathrm{B}' - \mathrm{B} - \sqrt{(\lambda + \mathrm{B}' - \mathrm{B})^2 + \Lambda_\mathrm{p}(\lambda) - \Lambda'_\mathrm{p}(\lambda)}\,. \qquad (21)$$

The data of Table 4 gives for the transition from the path of column 2 to 6 with (21): $\kappa > 5.3$. Therefore `\linepenalty` $= 10-6 = 4$ typesets three lines as mentioned above.

Table 4: Badness, penalties, and additional demerits of the line breaks for the five paths of example 3

		`\par via @@` (`*` is typeset)				
`@@`	Class	`*2`	`3`	`4`	`5`	`6`
`1`	l		86		86	
`2`	d	4		4		4
`3`	d		4			
`4`	l			57		
`5`	d				0	
`6`	d					6
`7`	d	7				
`(7)`	(d)		0	0	0	0
	$\mu =$	2	3	3	3	3
	B =	11	100	61	86	10
$\Lambda_\mathrm{p}(10) =$		485	9512	4785	9416	552

Summary. When the \linepenalty value is increased, TEX's line-breaking algorithm focuses more on the badness values. If a path exists in the network of line breaks that has the same number of lines but a lower sum of badness compared to the path selected with the default settings, that path might be chosen with the larger \linepenalty. This means that more breaks in mathematics and/or at positive \penalty commands and/or more hyphens and/or more stacks of hyphens and/or more visually incompatible lines are typeset and at least one of these items is increased.

If a path exists that uses fewer lines for the paragraph, this path can be selected with a large \linepenalty even if its sum of badness is higher than that of the paragraph with the default settings. Similarly a path can be selected that has more lines if \linepenalty stays positive but is made smaller than 10.

Negative values for \linepenalty typically create rather ugly paragraphs as TEX then prefers large badness values for the lines. This effect is not normally desirable for justified text.

6 \linepenalty versus \looseness

The TEXbook has an exercise in which the value 100 for the parameter \linepenalty is suggested as a replacement for a negative \looseness in an application to a single paragraph ([3], exercise 14.25). The reason refers to efficiency to "achieve almost the same result" if the user is not willing to pay the cost that a nonzero \looseness generates. (\looseness is explained in [3], pp. 103–104 or see Section 5 of [8] for an analysis of this parameter.)

Figure 2 shows that there are a lot of cases in which two lines are typeset instead of only one if \linepenalty = 100. And example 1 proves that the increase of \linepenalty can make a paragraph longer. Therefore this parameter might not only fail to reduce the number of lines it might be counterproductive. Although passes can have different numbers of lines for the shortest path, with a small enough negative \looseness a paragraph can never get more lines than it has in the earliest pass that typesets it if \pretolerance $\leq$ \tolerance, as the paths of the first pass are part of the network of line breaks of the second pass.

Inequality (19) can be used to determine in which cases a \linepenalty of 100, i.e., $\kappa = 90$, can be successful in general. Here the plain TEX values are used:

$$\begin{aligned} & 90^2 + 2 \cdot 90(10 + \mathrm{B} - \mathrm{B}') > \Lambda'_{\mathrm{p}}(10) - \Lambda_{\mathrm{t}}(10) \\ \iff \quad & 9900 - 180(\mathrm{B}' - \mathrm{B}) > \Lambda'_{\mathrm{p}}(10) - \Lambda_{\mathrm{t}}(10). \end{aligned}$$

Therefore the difference between the sum of badness values must be less than $55 = 9900/180$, but of course it must often be much smaller as the difference of the path demerits on the right hand side is positive and usually not very small.

Although Theorem 1 proves that the value 242 for \linepenalty acts like \looseness = −1 for a single line the scenario represents only a special case. For example, as noted in Section 3, the single line is always typeset by TEX's line-breaking algorithm in the first pass. In general these two parameters behave quite differently.

Second pass. This is *the* fundamental difference between these two parameters: \looseness will try hyphenation, i.e., the second pass, if it is not successful in the first.

Thus, hyphens might be introduced at the end of the lines if \looseness is used although no reduction of the number of lines is achieved.

Example 8: Description
Typeset the text of example 7 twice: first with plain TEX's defaults and second with \looseness = −1.

TEX output
Hi! TEX! Tell me: How is the following long word broken 'pneumonoultramicroscopicsilicovolcanoconiosis'? I am sure that you are an expert in hyphenation, right TEX? Or shall I ask Siri?
Hi! TEX! Tell me: How is the following long word broken 'pneumonoultramicroscopicsilicovolcanoconiosis'? I am sure that you are an expert in hyphenation, right TEX? Or shall I ask Siri? □

The change of \linepenalty never forces TEX's line-breaking algorithm to execute another pass. It uses the pass that is necessary to break the lines when \linepenalty = 10.

Example 9: Description
Typeset a paragraph twice: first with \linepenalty = 9799 and second with the default \linepenalty = 10 and \looseness = −1.

TEX output

A short text that cannot be typeset in two lines although looseness does it in the 2nd pass. A surprise! Or?

A short text that cannot be typeset in two lines although looseness does it in the 2nd pass. A surprise! Or? □

TEX typesets the text of the first paragraph in the first pass. The line-breaking algorithm cannot eliminate the third line in this pass. On the other hand this means that the first pass is a failure in TEX's view if \looseness = −1. But the second pass is a success: Although it also prefers three lines, there is a way to output only two. The demerits for the three line solution are 6926, those for the pair of lines 15773.

Different cost functions. This leads to the next difference: `\looseness` can choose a line-breaking solution that does not represent the shortest path in the network. This never happens for any setting of `\linepenalty`; it must pick the shortest path.

As `\looseness` has a different cost function to be optimized, penalties larger than −10000 and smaller than 10000 mark places that are as good as others for a line break.

Example 10: Description
A text with two penalties is typeset twice: first with `\linepenalty` = 9799 and second with `\looseness` = −1 and `\linepenalty` = 10.

TEX input

```
OK! Even 4-digit penalties, positive or negative,
are\penalty9999\ not important for looseness but
linepenalty obeys\penalty-9999\ them.
```

TEX output

OK! Even 4-digit penalties, positive or negative, are not important for looseness but linepenalty obeys them.

OK! Even 4-digit penalties, positive or negative, are not important for looseness but linepenalty obeys them.

Both paragraphs are typeset in the first pass. The `\linepenalty` must pick the shortest path in the network with the cost function of demerits and thus TEX typesets three lines. This cost function is not relevant for `\looseness` if the number of lines of the paragraph can be changed. Only if this is not possible does TEX select the shortest path in the current pass as usual. This means a parameter that does not inhibit some behavior does not count if `\looseness` can be successful.

Success rate. No single value of `\linepenalty` works for all paragraphs but `\looseness` is always successful if the paragraph can be typeset with fewer lines.

Example 4 shows a text that can be typeset in two or three lines; a pair is output if `\linepenalty` is set to 199. Example 5 typesets the text twice and it needs `\linepenalty` = 385 to keep four lines. Each additional repetition of the original text requires a larger `\linepenalty`:

number of copies	1	2	3	4	5	6	7
`\linepenalty` =	199	385	557	738	918	1098	1278

Of course, this is a constructed example, but nevertheless with 25 iterations the `\linepenalty` must be 4519. The next step, i.e., a paragraph with only 52 lines in the shortest form, cannot be typeset with a `\linepenalty` of 4541 anymore and it would need 4599 if the progression continues as before. It violates TEX's limit for the total demerits, see (4), and the text is not typeset correctly. With the default `\linepenalty` together with `\looseness` = −1 no problem occurs.

Note: The large value 9799 for `\linepenalty`, which was used in the last two examples, can be applied for paragraphs with at most ten lines. In order to demonstrate certain effects in examples its usage is needed, but such a large value would not be used for normal copy.

Quality of output. If the number of lines of a paragraph cannot be lowered then `\looseness` still tries to find a line-breaking solution that avoids visually incompatible lines and stacks of hyphens, i.e., it obeys the additional demerits `\adjdemerits` and `\doublehyphendemerits` if possible.

But Section 5 shows that `\linepenalty` trades the smaller sum of badness for penalties and additional demerits. This means that a text that contains no math, no explicit `\penalty` command, and no explicit hyphens must get visually incompatible lines in a first pass if TEX changes the line breaks.

In the second pass `\linepenalty` considers hyphens as `\looseness` does. The latter treats the parameters `\double...` and `\finalhyphendemerits` as usual while the former parameter trades them like `\adjdemerits` for a smaller sum of badness values. That is, TEX not only might typeset more hyphens, but also there might be more visually incompatible lines, more stacks of hyphens, and more hyphenated penultimate lines.

Efficiency. The advantage of increasing the parameter `\linepenalty` instead of using `\looseness` is that `\linepenalty`'s value is always added in the code of the line-breaking algorithm — even if it is zero. A nonzero `\looseness` invokes otherwise unused code and thus slows the algorithm down [4, §§ 873, 875]. As expected, this loss in efficiency is hardly noticed in normal copy with modern equipment.

My outdated computer from 2011 (a 1.8 GHz Dual-Core i7) with my own TEX installation shows the following factors by which the runtime increases. The reference value 1 is used for the time needed by plain TEX to typeset the two lines of example 4 with `\looseness` = 0.

copies	`\looseness` = 0	`\looseness` = −1
1	1	1
100	3	11
225	6	100

But even the abnormally long paragraph of the last case with 450 lines needs only one second to get typeset if `\looseness` = −1.

`\looseness` can be positive. A value larger than zero for the parameter `\looseness` tries to make the paragraph longer.

Negative values for `\linepenalty` usually typeset low quality paragraphs. This was proved in Section 5. Thus it is not a good idea to use them except when they would lengthen paragraphs; Section 5 shows that values < 10 can be succesful.

Example 3 uses the value 2 for `\linepenalty` to typeset three instead of two lines. It happens that the `\linepenalty` can be -2 (but not -3).

Example 11: Description
Typeset example 3 with `\linepenalty` $= -2$.

TEX output
When you start to count where do you start? With zero or with one? Hmm, I start at 1! A CS nerd uses a 0, or? ∎

The next example uses a small positive and a high negative value for `\linepenalty`. But even in a forced second pass (used by `\looseness` $= 1$) these values do not make TEX typeset an additional line.

Example 12: Description
Typeset a paragraph four times: first with plain TEX's settings, second with `\looseness` $= 1$, third in a forced second pass with `\linepenalty` $= 1$, and finally, still in the second pass, with a `\linepenalty` of -9999.

TEX output

This is a short paragraph and two words can have a hyphen in it. The rest of the text is made up of short words only. Well, I think the first sentence is wrong. Wait then one more must be wrong. Two are wrong.

This is a short paragraph and two words can have a hyphen in it. The rest of the text is made up of short words only. Well, I think the first sentence is wrong. Wait then one more must be wrong. Two are wrong.

This is a short paragraph and two words can have a hyphen in it. The rest of the text is made up of short words only. Well, I think the first sentence is wrong. Wait then one more must be wrong. Two are wrong.

This is a short paragraph and two words can have a hyphen in it. The rest of the text is made up of short words only. Well, I think the first sentence is wrong. Wait then one more must be wrong. Two are wrong. ∎

Summary. TEX must work harder if `\looseness` is negative but a large value of `\linepenalty` is not a replacement. A large value for the parameter `\linepenalty` might even increase the number of lines output for a paragraph.

- If a paragraph can be typeset with fewer lines than TEX's default settings produce then

`\looseness` $= n$ for some $n \leq -1$ is successful;

`\linepenalty` > 10 might be successful but only if the pass has not to be changed; otherwise the paragraph is treated like an unsuccessful case.

- If fewer lines for the paragraph are impossible

`\looseness` tries the final pass and thus might insert hyphens but outputs the shortest path;

`\linepenalty` outputs the (new) shortest path that might now have lines with lower badness, but then more breaks in mathematics or at positive `\penalty` commands or more hyphens or more stacks of hyphens or more visually incompatible lines are used.

Both parameters might have a "negative impact" on paragraphs that cannot be shortened but the outcome with a large `\linepenalty` seems to be worse. Its only advantage is that it does not change the pass and that it obeys other TEX parameters.

References

[1] David Bausum, *TEX Reference Manual*, Norwell, Massachusetts: Kluwer Academic Publishers, 2002 `tug.org/utilities/plain/cseq.html#linepenalty-rp`

[2] 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 [7], 67–155

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

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

[5] Donald E. Knuth, "The Errors of TEX", *Software — Practice and Experience* **19** (1989), 607–685; reprinted as Chapter 10 in [6], 243–291

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

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

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

[9] Udo Wermuth, "The optimal value for `\emergencystretch`", *TUGboat* **38**:1 (2017), 64–86 `tug.org/TUGboat/tb38-1/tb118wermuth.pdf`

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

Errata for previous articles. Here are two corrections for errors that are not merely typographical.

In [8], p. 365, left column, the `\hsize` of the books *The Art of Computer Programming* by D. E. Knuth should be 348 pt.

In [9], p. 75, left column no. 10, the value of `t` is wrong; the stated value represents 2`t`.