
On differences in paragraph length between \TeX 's first and second pass

Udo Wermuth

Abstract

This article looks at examples of straight-text paragraphs that are typeset with a different number of lines in \TeX 's first and second pass. Among other results it is shown that the difference is unbounded if the paragraph length can be arbitrarily large.

1 Introduction

It's well known that \TeX can execute two completely independent passes when it has to find the best line breaks in a paragraph. The second pass is tried only if the first is switched off or fails to typeset the text acceptably with its given parameter set [1, p. 96].

As the passes are independent, the number of lines that are found to give the best typeset result for a paragraph can be different [2, p. 369]. Let's name the number of lines in the first pass n_1 and in the second n_2 , if they exist. We call the difference of these numbers in this article d , i.e., $d := n_1 - n_2$. But we might not see n_1 even if it exists: When we use a nonzero \looseness , $l \neq 0$, and \TeX cannot get $n_1 + l$ lines, it executes the second pass and attempts to obey l with n_2 . We get a value in the interval formed by the numbers n_2 and $n_2 + l$ [1, ex. 14.25] if \emergencystretch equals 0pt [1, p. 107]. Otherwise, \TeX tries a third pass if we do not get $n_2 + l$ lines in the second [2, p. 370].

The difference d is not zero for all paragraphs. For instance, we can have $d = 1$, i.e., the first pass creates one more line than the second, or we can have $d = -1$, i.e., the second pass creates one more line than the first; see [2, p. 369] for examples. Of course, human curiosity gives us at once some questions. Can d become any integer, or are there numbers $b_1 < 0$, $b_2 > 0$ so that $b_1 \leq d \leq b_2$?

We have to expect that a paragraph with, say, $d = 20$ might have more than several hundred lines, i.e., something that rarely occurs in normal use. But smaller d can occur in practice and help to explain certain effects. For example, on page 336 of *A plain \TeX primer* (OUP, 1992), Malcolm Clark writes "It is said that by removing a word in a paragraph it is possible to increase the overall length of the text." His explanation of this scenario uses non-default settings for some plain \TeX parameters and is linked to page breaking. In this article we will see paragraphs that increase their lengths by one or more lines because one character is deleted.

Udo Wermuth

doi.org/10.47397/tb/46-3/tb144wermuth-parpass

Contents. Sections 2 and 3 introduce the methods and tools that are used to demonstrate the results in the next sections.

In the examples up to and including section 6 we use plain \TeX with its default settings; only the \hspace and the main font are changed, and sometimes the \pretolerance is set to -1 to force \TeX to execute the second pass. In section 4 we show that there is no $b_2 > 0$ such that $d \leq b_2$, i.e., the difference has no upper bound. Section 5 proves that there is no lower bound. That the removal of a single character can change the number of lines that \TeX uses to typeset a paragraph is shown in section 6.

Sections 7 and 8 check what happens with a zero \parfillskip or a nonzero \looseness . And section 9 analyzes the features of the second pass independently by either setting \tolerance to 100 or \hyphenpenalty to 10000. The last section contains a few remarks and a summary.

2 Repetition

First, a few facts about \TeX 's line-breaking procedure. The program finds the possible line breaks that obey the current parameter set, for example, the tolerance for stretching spaces between words, by considering a numeric value called *demerits* [1, p. 94]. It assigns demerits to each possible output line whose end is marked with a *feasible breakpoint* [1, p. 99]. Later it picks a set of feasible breakpoints so that the sum of the line demerits and *additional demerits* [1, p. 98] is a minimum for the paragraph.

We are interested in the *existence* of arbitrarily long paragraphs fulfilling a given requirement. But we are not interested in the *contents* of such paragraphs. Therefore, the long paragraphs are constructed by repeating a text several times. That is, we define a text block t_b and repeat it. We start with one or more copies of t_b and define a text block for the repetition. This *repetition block* is usually not simply t_b but a combination of several copies of t_b .

We have to prove that the repetition allows us to construct paragraphs fulfilling the stated requirement; otherwise it is not possible to know that they build an infinite sequence, although a few initial results seem to start one. So I add two conditions to the constructed paragraphs that must be kept by the repetition. One is about their endings, the other about the repetition block's impact on them. This allows us to prove a theorem about the whole construction that guarantees an infinite sequence.

First definition. We say that two blocks of text have the *same ending* if the last two lines of the blocks typeset by \TeX have the same contents and

the same badness, penalty, and demerits values. In \TeX 's language it means that their traces obtained with the setting `\tracingparagraphs=1` have, for the last two output lines, identical text lines and identical \@ -lines for the chosen breakpoints except for the \@@ -numbers, i.e., the identification numbers for feasible breakpoints; the number of unused \@ -lines might also differ. Moreover, the blocks might have a different number of total lines and thus a different number of feasible breakpoints and different total demerits. (Details about the data in this trace can be found in [1, pp.98–99] or [2, section 3].)

The important point is that both final breakpoints refer to breakpoints that make the last lines identical. Each of them must refer to a breakpoint such that the two penultimate lines become identical including additional demerits. With identical contents the badness values and the penalties at the line breaks are identical, i.e., the main contents of the \@ -lines must agree. Thus, in the chosen \@@ -lines (i) the change of the line numbers and (ii) the fitness classes, i.e., the number after the period, must agree.

Second definition. We define a text of n lines, t_i , and a repetition block, t_r , that each consists of several copies of a text block t_b separated by a space. We say t_r has a *limited impact on t_i* if t_i and $t_i \sqcup t_r$ have an identical first line. Using p for the pass, the number of changed lines in t_i is named $L_p(t_i, t_r) := n - k$ if the first k lines agree, k maximal. The remaining text of t_i after line k is called $T_p(t_i, t_r)$.

For example, t_i might contain the lines $l_1, l_2, \dots, l_{n-1}, l_n$ and $t_i \sqcup t_r$ has l'_1, l'_2, \dots, l'_m in the first pass. When the repetition block is added to t_i the contents of line n might change and thus $l_n \neq l'_n$. There just might be new text from the repetition block or l_{n-1} gives some material to form the new l'_n . In the second case l_{n-1} also becomes a new line l'_{n-1} , etc. But $l_j = l'_j$ for $1 \leq j \leq n - L_1(t_i, t_r)$ if t_r has a limited impact on t_i .

A theorem. Let's assume that we have two blocks of text, called t_0 and t_1 , that both consist of repeated copies of the same text t_b . Moreover,

- (i) t_0 and $t_0 \sqcup t_1$ have the same ending and
- (ii) t_1 has a limited impact on t_0 and also on $t_0 \sqcup t_1$ with $L_p(t_0, t_1) = L_p(t_0 \sqcup t_1, t_1)$.

Then $t := t_0 \sqcup t_1 \sqcup t_1$ has the same ending as t_0 and t_1 the same limited impact on t , i.e., $L_p(t_0, t_1) = L_p(t, t_1)$. Moreover, t is typeset by \TeX in the first pass if t_0 and $t_0 \sqcup t_1$ are typeset in the first pass.

Proof. We add to t_0 and $t_0 \sqcup t_1$ the same text block t_1 . That means that the last lines of t_0 and $t_0 \sqcup t_1$ can receive the same text from the beginning of t_1

to build the next feasible breakpoint. We know how t_0 reacts on t_1 and this must happen with $t_0 \sqcup t_1$ too as by (i) the texts have the same ending. Thus all feasible breakpoints in t_1 after t_0 are also present in the block t_1 after $t_0 \sqcup t_1$. This also proves that $t_0 \sqcup t_1 \sqcup t_1$ can be typeset in the same pass as $t_0 \sqcup t_1$.

By (ii) and the initial condition about repeated copies of t_b in t_0 and t_1 we must have $T_p(t_0, t_1) = T_p(t_0 \sqcup t_1, t_1)$. \TeX cannot distinguish the two paragraphs in their last $L_p(t_0, t_1) + 1$ lines and only those influence the line breaking of the repetition block t_1 , including additional demerits. Thus, \TeX produces for the last t_1 in t the same output as for t_1 in $t_0 \sqcup t_1$. That is, t has the same ending as $t_0 \sqcup t_1$ and $L_p(t, t_1)$ must equal $L_p(t_0 \sqcup t_1, t_1)$. \square

With this theorem an infinite repetition chain can be established. Assume we have $t_i = t_b \sqcup \dots \sqcup t_b$ and a repetition block $t_r = t_b \sqcup \dots \sqcup t_b$. First, show that the theorem's conditions are fulfilled if $t_0 = t_i$ and $t_1 = t_r$. Then apply the theorem to $t_0 \sqcup t_1$ and $t_0 \sqcup t_1 \sqcup t_1$, etc.

Verification. Of course, it's somewhat complicated to verify the conditions for the theorem using the trace data written by `\tracingparagraphs=1`. One can simplify the check through a visual comparison of the typeset output; it's a sufficient condition.

Two paragraphs have the same ending if the last three lines agree. We must check three and not two lines to avoid any influence of \TeX 's additional demerits on the penultimate line.

The other condition, $L_p(t_0, t_1) = L_p(t_0 \sqcup t_1, t_1)$, can be directly checked through the typeset results. We go back from t_0 's last line in $t_0 \sqcup t_1$ till a line break is found that occurs in t_0 too. All previous line breaks must also agree. Next, one must verify that this line break is seen in $t_0 \sqcup t_1 \sqcup t_1$ too.

Note: Most paragraphs that are used to prove the results consist of many lines; thus, they cannot be shown. In section 4 I present a verification with traces and an excerpt of typeset lines. To show these text snippets I use the `\hsize` of this journal's column, i.e., 225 pt with the font `cmr9`; the paragraphs are typeset without indentation. But most often, I only provide some data that indicate that the scenario fulfills the conditions of the theorem.

3 The test environment

Macros help to make \TeX execute a text with the correct parameter set. They correctly count the produced lines. They ensure that the same text is executed in both the first and the second pass. And they guarantee a consistent naming scheme for the results.

Macros to execute examples. \TeX 's internal parameter `\prevgraf` stores the number of lines that \TeX used to typeset the most recently completed (straight-text) paragraph [1, p. 103].

In the test environment we enclose the text that has to be typeset by two macros: `\PBbegin` and `\PBend`. (The names of this article's macros start with "PB" except for the shortcut `\st`.)

```
\hsize=225pt \font\ninerm=cmr9
\newcount\PBlength % stores \prevgraf's value
% to be used as
% \PBbegin<text>\PBend<integer>:<name>.
% where <text> is typeset in a \vbox
% with \pretolerance set to <integer> and
% <name> becomes a macro containing \prevgraf
\def\PBbegin{\setbox0=\vbox\bgroup\ninerm
  \def\st{\space\the}\noindent}
\def\PBend#1:#2.{% #1: value for \pretolerance,
% #2: name for macro to store number of lines
  \pretolerance=#1\par \global\PBlength=\prevgraf
  \expandafter\xdef\csname #2\endcsname
    {#2: \the\PBlength}\egroup}
```

The names of macros for results begin with "PiB" or "PiiB" according to the designated pass. Thus, `\PBbegin Hello world!\PBend-1:PiiBexample` typesets the text "Hello world!" with a suppressed first pass, i.e., \TeX uses the second pass, and stores the number `\prevgraf` in the macro `\PiiBexample`. We get with `\show` the expected result.

```
> \PiiBexample=macro:
->PiiBexample: 1.
```

Remember that we can force the use of the second pass with the value `-1` for `\pretolerance`. But the value `100`, i.e., plain \TeX 's default value, is only a starting value. \TeX switches to the second pass if the text cannot be broken acceptably with respect to the given parameters in the first; then $n_1 := n_2$.

As we want to compare the results of the first and second pass, it makes sense to have macros that execute \TeX twice for a text. The number of 'i's between 'P' and 'B' signals the pass that is tried first. The number of the text copies is given at the end, also using roman numerals.

```
\newcount\PiBlength \newcount\PiiBlength
\def\PBpasses(#1)#2.{% #1: <text>; #2: number;
% should equal the number of copies of t_b in #1
  \PBbegin #1\PBend100:PiB\romannumeral#2.\par
  \global\PiBlength=\PBlength
  \PBbegin #1\PBend-1:PiiB\romannumeral#2.\par
  \global\PiiBlength=\PBlength
  \PBlines(\romannumeral#2)\ignorespaces}
\def\PBlines(#1){% #1: roman numeral or text
  \line{\csname PiB#1\endcsname\hfil
    \hbox to 140pt{\csname PiiB#1\endcsname\hfil
      \hbox to 40pt{\PBlength=\PiBlength
```

```
\advance\PBlength by -\PiiBlength
$d=\number\PBlength$\hss}}}
```

For example,

```
\PBpasses>Hello world!)1.
```

outputs one line of text with the results obtained in both passes and the difference of the produced lines.

```
PiBi: 1          PiiBi: 1          d = 0
```

Registers to store examples. Usually the text is stored in one or more token registers. For example, assume we have one register with the name `\PBtb` and one with `\PBtr` that contains `\PBtb` twice, separated by a space. Then we can enter

```
\PBbegin\the\PBtb\space\the\PBtr\PBend100:PiBiii
so that  $\TeX$  builds a paragraph from both registers with a space between their contents. (Later we use \st for \space\the; see the code of \PBbegin.) The whole construction is named with another roman numeral. The one after 'B' shows that \PBtb is used thrice. But as mentioned earlier we want to execute the text in both passes of  $\TeX$ . With this fail-safe input everything is executed and named according to our scheme: \PBpasses(\the\PBtb\st\PBtr)3.
```

```
\newtoks\PBtb % the text
\newtoks\PBtr % the repetition block
\newtoks\PBts % the second repetition block
\newtoks\PBtt % the third repetition block
\newtoks\PBte % for error situations
\newtoks\PBti % for intermediate calculations
```

4 Difference d has no upper bound

This means the first pass is able to create paragraphs that are longer by any given number of lines than the same paragraph typeset in the second pass.

Text and repetition blocks. We take a rather long text and a repetition block with two copies. Here are the assignments to the token register.

```
\PBtb={In a day or two one has a lot to do to
find complex hyphenation patterns that help to
make a few lines of text like this one. I saw a
claim that one can find an example text that can
be broken in the 2nd pass with two lines less
compared to the 1st pass of a  $\TeX$  run. Some
believe that then line by line the 2nd pass has
more characters in each of its lines so that it
makes the paragraph quickly shorter. But think
about it! How slow it gains 1pt after the other!
It saves just one third of a line in ten or more
lines of text. So to reach the reduction of more
than a text line, one must build a paragraph of
many lines, I say, more than thirty. Shall one
try to construct such a paragraph? Well, it is
a time-consuming task, I think, a task that I
should do before I state it is possible, but no
```

proof is given. The interaction of `\TeX`'s parameters seems to be so tricky that it's good to give an example. The 1st pass behaves often like a diva: One wrong word and it stops! Leaves the stage! Poor 2nd pass must do the job! It is more than annoying. So each word must be chosen carefully to make the 1st pass happy. And it is very important for the progress of the example that all selected words help the 2nd pass to gain ground! Yes, each line shall make a step in the right direction. Or it is a useless, a wasted step. A line that isn't needed to reach the goal is constructed. But the steps are tiny. It's slow, a fight for every point as I wrote earlier. How is it going? OK, two thirds are done. Or isn't it? It is now a very long paragraph but it has only one hyphen in its first line. Well, one is required, of course, but such a dash is bad as it needs space. It's a new symbol in the 2nd pass, so progress is slowed down. It's not wise to see many hyphens in the text of the 2nd pass. I try not to use one! Up to now it worked well. As the task is nearly done I expect to avoid a 2nd one. In one of the next lines it happens that the 2nd pass sets the text a line shorter. It is good to end this! I am sure you find it boring. I do. Let's come to the end---let's finish this! Now the easy part starts. I need, say, another hyphen in the 2nd pass. Please, only one hyphen. Just one hyphen! Oops!}

```
% repetition blocks with 2, 4, and 6 copies
\Btr={\the\Btb\st\Btb}
\Bts={\the\Btr\st\Btr}
\Btt={\the\Btr\st\Bts}
```

`\TeX` needs 41 lines to typeset the above text. In the second pass `\TeX` hyphenates the word “hyphenation” and ends the first line with this hyphen. Then it needs only 39 lines to typeset the text. Here is the (partial) result of the first pass; the first three and the last four lines of the 41 lines are shown. The typeset text does not provide new insights but we later discuss some of the following line breaks.

In a day or two one has a lot to do to find complex hyphenation patterns that help to make a few lines of text like this one. I saw a claim that one can find an
 [34 lines not shown]
 boring. I do. Let's come to the end—let's finish this! Now the easy part starts. I need, say, another hyphen in the 2nd pass. Please, only one hyphen. Just one hyphen! Oops!

The last line has one short sentence: “Oops!” Otherwise the lines contain 8–14 words: (8, 9, 10, 11, 12, 13, 14) words occur (1, 2, 6, 13, 10, 6, 2) times. Thus, in total there are 456 words. 40 completely filled lines for 455 words result in an average

of 11.375 words/line. The lines have 50–58 characters and spaces; the last line contains five characters. The characters (‘.’, ‘!’, ‘?’) appear (26, 9, 3) times, i.e., there are 38 sentences.

`\TeX` breaks the text in the second pass differently. As explained above, the first line ends with a hyphenated word directly indicating a second pass:

In a day or two one has a lot to do to find complex hyphenation patterns that help to make a few lines of text
 [34 lines not shown]

Let's come to the end—let's finish this! Now the easy part starts. I need, say, another hyphen in the 2nd pass. Please, only one hyphen. Just one hyphen! Oops!

Results. We build several paragraphs and run `\TeX` on each of them in both passes.

```
\PBpasses(\the\PBtb)1.
\PBpasses(\the\PBtb\st\PBtr)3.
\PBpasses(\the\PBtb\st\PBts)5.
\PBpasses(\the\PBtb\st\PBtt)7.
\PBpasses(\the\PBtb\st\PBtt\st\PBtr)9.
\PBpasses(\the\PBtb\st\PBtt\st\PBts)11.
\PBpasses(\the\PBtb\st\PBtt\st\PBtt)13.
\PBpasses(\the\PBtb\st\PBtt\st\PBtt\st\PBtr)15.
\PBpasses(\the\PBtb\st\PBtt\st\PBtt\st\PBts)17.
\PBpasses(\the\PBtb\st\PBtt\st\PBtt\st\PBtt)19.
```

This is the result: left is n_1 , in the middle n_2 . It shows that $d := n_1 - n_2$ can become 2, 3, ..., 11.

PiBi: 41	PiiBi: 39	$d = 2$
PiBiii: 121	PiiBiii: 118	$d = 3$
PiBv: 201	PiiBv: 197	$d = 4$
PiBvii: 281	PiiBvii: 276	$d = 5$
PiBix: 361	PiiBix: 355	$d = 6$
PiBxi: 441	PiiBxi: 434	$d = 7$
PiBxiii: 521	PiiBxiii: 513	$d = 8$
PiBxv: 601	PiiBxv: 592	$d = 9$
PiBxvii: 681	PiiBxvii: 671	$d = 10$
PiBxix: 761	PiiBxix: 750	$d = 11$

To prove that the sequence for d continues we have to look at `\tracingparagraphs`' trace of, say, `\PiBi` and `\PiBiii` as well as `\PiiBi` and `\PiiBiii` and check that these pairs have the same endings and repetition blocks with limited impact. Figure 1 shows the details of the trace for `\PiBi`'s ending. The trace for `\PiBiii` is very similar, except that its @@-numbers are larger by 970, the line numbers by 80, and the demerits by 42384; see Figure 2.

The last lines in these traces, i.e., line 41 and line 121, contain the word “Oops!” as the last feasible breakpoints @@239 and @@1209 use the previous breakpoints: @@238 and @@1208, respectively. These two breakpoints refer to the breakpoints in front of the penultimate lines. Thus, the penultimate lines start after the breakpoints @@228 and @@1198, respectively. They both contain “the 2nd pass. Please,

```

@@228: line 39.2 t=40429 -> @@215
the
@ via @@217 b=86 p=0 d=9216
@ via @@218 b=86 p=0 d=9216
@@229: line 39.1 t=55185 -> @@217
2nd
@ via @@217 b=0 p=0 d=100
@ via @@218 b=0 p=0 d=100
@@230: line 39.2 t=46069 -> @@217
pass.
@ via @@219 b=6 p=0 d=256
@ via @@220 b=81 p=0 d=8281
@@231: line 39.2 t=45798 -> @@219
Please,
@ via @@221 b=48 p=0 d=3364
@@232: line 39.1 t=55935 -> @@221
only
@ via @@221 b=35 p=0 d=2025
@ via @@222 b=0 p=0 d=100
@@233: line 39.2 t=52623 -> @@222
@@234: line 39.3 t=54596 -> @@221
one
@ via @@223 b=7 p=0 d=289
@@235: line 39.2 t=49586 -> @@223
hyphen. Just
@ via @@225 b=0 p=0 d=100
@ via @@226 b=0 p=0 d=100
@@236: line 39.2 t=48863 -> @@226
one
@ via @@227 b=70 p=0 d=6400
@@237: line 40.1 t=46654 -> @@227
hyphen!
@ via @@228 b=46 p=0 d=3136
@ via @@229 b=14 p=0 d=576
@@238: line 40.3 t=43565 -> @@228
Oops!
@ \par via @@230 b=0 p=-10000 d=100
@ \par via @@231 b=0 p=-10000 d=100
@ \par via @@232 b=0 p=-10000 d=100
@ \par via @@233 b=0 p=-10000 d=100
@ \par via @@234 b=0 p=-10000 d=100
@ \par via @@235 b=0 p=-10000 d=100
@ \par via @@236 b=0 p=-10000 d=100
@ \par via @@237 b=0 p=-10000 d=100
@ \par via @@238 b=0 p=-10000 d=100
@@239: line 41.2- t=43665 -> @@238

```

Figure 1: `\tracingparagraphs`' trace of the last two lines of `\PiBi`

only one hyphen. Just one hyphen!" as can also be verified by the earlier shown typeset output.

Here is an excerpt showing the lines at the end of `\PBtb` when words are added to its start, i.e., when the repetition block is added:

```

In a day or two one has a lot to do to find complex
[several lines not shown]
boring. I do. Let's come to the end—let's finish this!
Now the easy part starts. I need, say, another hyphen

```

Udo Wermuth

```

@@1198: line 119.2 t=82813 -> @@1185
the
@ via @@1187 b=86 p=0 d=9216
@ via @@1188 b=86 p=0 d=9216
@@1199: line 119.1 t=97569 -> @@1187
...
@@1208: line 120.3 t=85949 -> @@1198
Oops!
@ \par via @@1200 b=0 p=-10000 d=100
...
@ \par via @@1208 b=0 p=-10000 d=100
@@1209: line 121.2- t=86049 -> @@1208

```

Figure 2: Excerpt of `\tracingparagraphs`' trace of the last two lines of `\PiBiii`

in the 2nd pass. Please, only one hyphen. Just one hyphen! Oops! In a day or two one has a lot to do to
[several lines not shown]

The breakpoint after the word “this!” establishes $L_1(t_b, t_r) = 3$; compare it to the last lines of `\PBtb` in the first pass shown earlier. I omit showing that we also have $L_1(t_b \sqcup t_r, t_r) = 3$.

These two verification steps (same endings and repetition with limited impact) must be performed for the typeset results of the second pass. I omit showing the traces that prove that the required conditions hold in this pass too. I give only a short summary: Execute the paragraphs in the second pass with `\tracingparagraphs=1` to see that the trace that it writes contains 32 breakpoints for the last two lines of `\PiBi` and `\PiBiii`. These parts of the traces consist of 127 and 131 lines, respectively, and agree in all important aspects. For these paragraphs we get $L_2(t_b, t_r) = L_2(t_b \sqcup t_r, t_r) = 1$.

5 Difference d has no lower bound

Or with other words, a paragraph typeset in \TeX 's second pass can become any number of lines longer than the same paragraph typeset in the first pass.

Text and repetition blocks. This time we take a text and repeat it three times before we add the repetition block that consists of five copies of the text. Here are the used token registers; `\PBti` contains the three copies of `\PBtb`.

```

\PBtb={A ‘B’, a ‘T’, an ‘L’, then an ‘M’ and an
‘N’: ‘Beatlemania’ consists of these consonants;
each letter occurs once. All other letters are,
of course, vowels: three ‘a’, two ‘e’, and one
‘i’. The total number of letters
is the ‘foolish eleven.’}
% create a block with 3 copies of the text
\PBti={\the\PBtb\st\PBtb\st\PBtb}
% repetition blocks with 5, 10, and 15 copies
\PBtr={\the\PBti\st\PBti\st\PBti}

```

```
\PBts={\the\PBtr\st\PBtr}
\PBtt={\the\PBtr\st\PBts}
```

`\PBtb`, or t_b , is typeset in the first pass in four lines.

A ‘B’, a ‘T’, an ‘L’, then an ‘M’ and an ‘N’: ‘Beatlemania’ consists of these consonants; each letter occurs once. All other letters are, of course, vowels: three ‘a’, two ‘e’, and one ‘i’. The total number of letters is the ‘foolish eleven.’

There are 44 words in total as the lines have 13, 9, 11, and 11 words. The average is 11 words/line. There are 59, 58, 61, and 61 characters and spaces in the four lines. The character ‘.’ occurs three times.

In the first pass the block t_i is a simple stack of the above shown t_b , i.e., three copies have twelve lines. In the second pass we get for t_i the following line breaks:

A ‘B’, a ‘T’, an ‘L’, then an ‘M’ and an ‘N’: ‘Beatlemania’ consists of these consonants; each letter occurs
[8 lines not shown]
each letter occurs once. All other letters are, of course, vowels: three ‘a’, two ‘e’, and one ‘i’. The total number of letters is the ‘foolish eleven.’

Results. We create several paragraphs.

```
\PBpasses(\the\PBti)3.
\PBpasses(\the\PBti\st\PBtr)8.
\PBpasses(\the\PBti\st\PBts)13.
\PBpasses(\the\PBti\st\PBtt)18.
\PBpasses(\the\PBti\st\PBtt\st\PBtr)23.
\PBpasses(\the\PBti\st\PBtt\st\PBts)28.
\PBpasses(\the\PBti\st\PBtt\st\PBtt)33.
\PBpasses(\the\PBti\st\PBtt\st\PBtt\st\PBtr)38.
\PBpasses(\the\PBti\st\PBtt\st\PBtt\st\PBts)43.
\PBpasses(\the\PBti\st\PBtt\st\PBtt\st\PBtt)48.
```

They give us the values $-1, -2, \dots, -10$ for d :

PiBiii: 12	PiiBiii: 13	$d = -1$
PiBviii: 32	PiiBviii: 34	$d = -2$
PiBxiii: 52	PiiBxiii: 55	$d = -3$
PiBxviii: 72	PiiBxviii: 76	$d = -4$
PiBxxiii: 92	PiiBxxiii: 97	$d = -5$
PiBxxviii: 112	PiiBxxviii: 118	$d = -6$
PiBxxxiii: 132	PiiBxxxiii: 139	$d = -7$
PiBxxxviii: 152	PiiBxxxviii: 160	$d = -8$
PiBxliii: 172	PiiBxliii: 181	$d = -9$
PiBxlviii: 192	PiiBxlviii: 202	$d = -10$

Again, I omit the trace data for the endings of `\PiBiii`, `\PiiBiii`, etc. We find $L_1(t_i, t_r) = 0$, i.e., each t_r adds twenty lines without an impact on the previous lines. For the second pass we get $L_2(t_i, t_r) = 1$.

6 Deletion of one character changes paragraph length by more than one line

The removal of a word in a text might change the number of lines in unexpected ways. But the term

“word” is not very precise. Does the used method depend on the length of the word? Must the length be a certain percentage of the `\hsize`? And so on. Therefore I decided to reduce the change to a single character. It might be a text character, a space, a discretionary break, e.g., `\-`, or an active character, e.g., the tie.

Deletion reduces the length by more than one line. First, we show that the deletion of a single character can shorten a paragraph’s length by two lines. That a text needs fewer lines if a character is deleted might be expected. That `TEX` is able to save two lines appears as a bit of a surprise.

We delete the last character of a text. With plain `TEX`’s defaults a text typeset in the first pass continues to be typeset in the first pass. Therefore we need a text that requires the second pass and as soon as the character is deleted `TEX` finishes the typesetting in the first pass.

We mainly reuse the text of section 5 although the last repetition is changed in the last line.

```
\PBte={A ‘B’, a ‘T’, an ‘L’, then an ‘M’ and an
‘N’: ‘Beatlemania’ consists of these consonants;
each letter occurs once. All other letters are,
of course, vowels: three ‘a’, two ‘e’, and one
‘i’. The total number of letters
equals ‘mathematics’.% <<< this line is changed
```

This time we execute two similar texts in the first pass. At least, we ask `TEX` to try the first pass. In `\PiBok` it is able to typeset the text, for `\PiBbad` it switches to the second pass.

```
\PBbegin
\the\PBti\st\PBti\st\PBtb\st\PBte
\PBend100:\PiBok.\par
\PBbegin
\the\PBti\st\PBti\st\PBtb\st\PBte’% with error
\PBend100:\PiBbad.
```

The second text has a typo: At the end of the text there is a closing quote. We find “`\PiBok: 32`” and “`\PiBbad: 34`”. That is, we wrote a paragraph with 34 lines, see a typo, and after its fix a paragraph with 32 lines results.

Of course, the texts can be made longer by inserting copies of `\PBtb`. After inserting five more copies for the text of section 5 we see that `\PiBbad` reports 55 lines and after deleting the final quote only 52 lines remain. That means we get the numbers shown in section 5.

Deletion increases the length. We show that a text can become longer if a character is deleted. The text of section 4 is typeset in the first pass with 39 lines if we enter the text with “`hy\phenation`”. This is the only second-pass feature that is needed.

After defining `\PBti` like `\PBtb` of section 3 but with the described discretionary break, the call

```
\tracingparagraphs=1
\PBbegin\the\PBti\PBend100:PiBi.\par
```

gives the result “PiBi: 39”. The trace data written by `\tracingparagraphs` proves that `TeX` uses the first pass. Thus, the deletion of “\” in the new text increases the number of lines by 2 as the first pass in section 4 has 41 lines.

OK, some readers might find it unfair that a discretionary hyphen is deleted at a place that’s also a feasible breakpoint. The paragraph has 39 lines if the input contains “`complex hy phenation`”. The same result of 39 lines is obtained if we use the following tie: “`complex~hyphenation`”. In the first case we stay in `TeX`’s first pass, in the second case `TeX` switches to the second. Thus, the deletion of either the second space or of the tie extends the paragraph by two lines as `TeX` can stay or switch back to an error-free first pass, respectively.

And this case can be extended, too. Rebuild the original sequence of section 4 but replace the first text block by the one containing the tie; versions with “\” and space do not scale. `TeX` must use the second pass and, thus, the removal of the tie increases the paragraph’s length as the paragraph can then be typeset in the first pass.

7 Use zero `\parfillskip`

We’ve worked with the default parameter settings of `plain`. What results do we get if we change some? First, let’s look at `\parfillskip` [1, p. 100].

Difference between passes. Plain `TeX` assigns the value `0pt plus 1fil` to `\parfillskip` to allow `TeX` to finish the last line of a paragraph at any position within the given `\hsize`. With an assignment of `0pt`, `TeX` typesets the paragraphs as rectangles because our paragraphs have no indentation.

First, we use the input of section 4 and typeset it with the zero `\parfillskip`. The last three lines are in the passes identical. Here is the ending of the typeset output:

```
In a day or two one has a lot to do to find complex
[several lines not shown]
```

do. Let’s come to the end—let’s finish this! Now the easy part starts. I need, say, another hyphen in the 2nd pass. Please, only one hyphen. Just one hyphen! Oops! But the values of $L_1(t_b, t_r)$ and $L_2(t_b, t_r)$ increase to 18 and 5, respectively.

PiBi: 40	PiiBi: 39	$d = 1$
PiBiii: 120	PiiBiii: 118	$d = 2$
PiBv: 200	PiiBv: 197	$d = 3$

PiBvii: 280	PiiBvii: 276	$d = 4$
PiBix: 360	PiiBix: 355	$d = 5$
PiBxi: 440	PiiBxi: 434	$d = 6$
PiBxiii: 520	PiiBxiii: 513	$d = 7$
PiBxv: 600	PiiBxv: 592	$d = 8$
PiBxvii: 680	PiiBxvii: 671	$d = 9$
PiBxix: 760	PiiBxix: 750	$d = 10$

The values for the first pass in section 4 are all reduced by one line but an increasing sequence for the values of d remains. Next, we do the same check with the input of section 5.

PiBiii: 12	PiiBiii: 12	$d = 0$
PiBviii: 32	PiiBviii: 34	$d = -2$
PiBxiii: 52	PiiBxiii: 55	$d = -3$
PiBxviii: 72	PiiBxviii: 76	$d = -4$
PiBxxiii: 92	PiiBxxiii: 97	$d = -5$
PiBxxviii: 112	PiiBxxviii: 118	$d = -6$
PiBxxxiii: 132	PiiBxxxiii: 139	$d = -7$
PiBxxxviii: 152	PiiBxxxviii: 160	$d = -8$
PiBxliii: 172	PiiBxliii: 181	$d = -9$
PiBxlviii: 192	PiiBxlviii: 202	$d = -10$

This time only the first value for the second pass is changed from 13 to 12 so we keep the decreasing sequence for $d < -1$.

The ending of the first pass does not change. For $t_i \sqcup t_r$ we get in the second pass:

```
A ‘B’, a ‘T’, an ‘L’, then an ‘M’ and an ‘N’: ‘Beatle-
[30 lines not shown]
```

of these consonants; each letter occurs once. All other letters are, of course, vowels: three ‘a’, two ‘e’, and one ‘i’. The total number of letters is the ‘foolish eleven.’

Now we have $L_2(t_i \sqcup t_r, t_r) = 30$. But we still have $L_1(t_i, t_r) = 0$.

Difference between default and cleared value. But we might also concentrate on `\parfillskip`’s value and ask if there is an unbounded difference between the number of lines of a paragraph if it is once typeset with `plain`’s default value of `\parfillskip` and once with a zero `\parfillskip`. Unbounded differences become possible if `TeX` has to switch the pass if the `\parfillskip` is cleared.

We use the text of section 5 that has a completely filled last line in the first pass and destroy this by adding a short word.

```
\newtoks\PBta \PBta={Okay.}% add single word
```

The addition of the word forces `TeX` to switch to the second pass if `\parfillskip` is cleared.

As we do not compare results of `TeX`’s passes, we need to extend the naming convention: To signal the zero `\parfillskip`, a “Z” is added to the “PiB...” names. That is, we use the following code.

```
\let\PiBZlength=\PiiBlength % use better name
\def\PBpasses(#1)#2.{% #1: <text>; #2: number
```

```

\PBbegin #1\PBend100:PiB\romannumeral#2.\par
\global\PiBlength=\PBlength
\PBbegin \parfillskip=0pt\relax #1\PBend
100:PiB\romannumeral#2Z.\par
\global\PiBZlength=\PBlength
\PBlines(\romannumeral#2)\ignorespaces}
\def\PBlines(#1){% #1: roman numeral or text
\line{\csname PiB#1\endcsname\hfil
\hbox to 40pt{\PBlength=\PiBlength
\advance\PBlength by -\PiBZlength
$d_p=\number\PBlength$\hss}}}}

```

Note: The new difference is no longer $d = n_1 - n_2$ so it gets a new name: d_p .

Now, we can execute the paragraphs of section 5 with the added word, i.e., we run:

```

\PBpasses(\the\PBti\st\PBta)3.
\PBpasses(\the\PBti\st\PBtr\st\PBta)8.
\PBpasses(\the\PBti\st\PBts\st\PBta)13.
\PBpasses(\the\PBti\st\PBtt\st\PBta)18.
\PBpasses(\the\PBti\st\PBtt\st\PBtr\st\PBta)23.
\PBpasses(\the\PBti\st\PBtt\st\PBts\st\PBta)28.
\PBpasses(\the\PBti\st\PBtt\st\PBtt\st\PBta)33.
\PBpasses
(\the\PBti\st\PBtt\st\PBtt\st\PBtr\st\PBta)38.
\PBpasses
(\the\PBti\st\PBtt\st\PBtt\st\PBts\st\PBta)43.
\PBpasses
(\the\PBti\st\PBtt\st\PBtt\st\PBtt\st\PBta)48.

```

and get these results.

PiBiii: 13	PiBiiiZ: 13	$d_p = 0$
PiBviii: 33	PiBviiiZ: 34	$d_p = -1$
PiBxiii: 53	PiBxiiiZ: 55	$d_p = -2$
PiBxviii: 73	PiBxviiiZ: 76	$d_p = -3$
PiBxxiii: 93	PiBxxiiiZ: 97	$d_p = -4$
PiBxxviii: 113	PiBxxviiiZ: 118	$d_p = -5$
PiBxxxiii: 133	PiBxxxiiiZ: 139	$d_p = -6$
PiBxxxviii: 153	PiBxxxviiiZ: 160	$d_p = -7$
PiBxliii: 173	PiBxliiiZ: 181	$d_p = -8$
PiBxlviii: 193	PiBxlviiiZ: 202	$d_p = -9$

The application of the zero `\parfillskip` extends the length of the paragraph except in the first case. That happens because \TeX needs the second pass to obey the changed parameter value. The additional word is absorbed in the number of lines that we saw in the previous table. Thus, we get a decreasing sequence of differences.

The same technique can be tried with the text of section 4. Here a longer text can be used, at least, as long as the first pass is not able to reach the full measure with the last line.

```
\PBta={Okay, that is a good result.}
```

When we add this sentence to the paragraphs of section 4—in the way we extended before the text of section 5 with `\PBta`—we get these results.

PiBi: 41	PiBiZ: 40	$d_p = 1$
PiBiii: 121	PiBiiiZ: 119	$d_p = 2$
PiBv: 201	PiBvZ: 198	$d_p = 3$
PiBvii: 281	PiBviiZ: 277	$d_p = 4$
PiBix: 361	PiBixZ: 356	$d_p = 5$
PiBxi: 441	PiBxiZ: 435	$d_p = 6$
PiBxiii: 521	PiBxiiiZ: 514	$d_p = 7$
PiBxv: 601	PiBxvZ: 593	$d_p = 8$
PiBxvii: 681	PiBxviiZ: 672	$d_p = 9$
PiBxix: 761	PiBxixZ: 751	$d_p = 10$

Again, the middle column switches to the second pass and we get a sequence of increasing differences. (Strictly speaking, the theorem of section 2 is not applicable. But we could extend the theorem as `\PBta` has a calculable influence on the line breaks.)

8 Applying `\looseness`

Another parameter to use is `\looseness` [1, pp. 103–104], which was briefly mentioned in section 1.

Let’s execute the paragraphs used in sections 4 and 5 with a nonzero `\looseness`, $l \neq 0$. To indicate that `\looseness` is involved I add to the name (with a code change similar to section 7) either the ending “M” or “P” followed by its absolute value shown as a roman numeral. The uppercase letter represents the sign: M signals a negative and P a positive value.

For `\parfillskip` we used a fixed value but for `\looseness` each line has its own value. I take the value d from the subsections “Results” and reverse its sign: $l := -d$. That is, we want to increase or decrease by this number of lines in the first pass.

On the left we have the result for $l = 0$ and in the middle the one for $l_i = -d$ where d is taken from the corresponding result in the i th line of section 4. The new difference in the third column is again not $d = n_1 - n_2$ so it gets a new name: d_l .

PiBi: 41	PiBiMii: 39	$d_l = 2$
PiBiii: 121	PiBiiiMiii: 118	$d_l = 3$
PiBv: 201	PiBvMiv: 197	$d_l = 4$
PiBvii: 281	PiBviiMv: 276	$d_l = 5$
PiBix: 361	PiBixMvi: 355	$d_l = 6$
PiBxi: 441	PiBxiMvii: 434	$d_l = 7$
PiBxiii: 521	PiBxiiiMviii: 513	$d_l = 8$
PiBxv: 601	PiBxvMix: 592	$d_l = 9$
PiBxvii: 681	PiBxviiMx: 671	$d_l = 10$
PiBxix: 761	PiBxixMxi: 750	$d_l = 11$

\TeX successfully reduces the number of lines in the first pass except for the first line, which uses the second pass. With a trace by `\tracingparagraphs` this can be verified. Thus, in the first line $d_l = n_1 - n_2$ and otherwise $d_l = n_1 - (n_1 + l_i) = -l_i$.

The above needs a lot of resources. The trace of the last paragraph with 761 lines in the first pass

contains 8873 feasible breakpoints. The one with `\looseness=-11` that `TeX` writes to create 750 lines in the first pass reports a formidable 92314.

The theorem of section 2 is applicable as `TeX` reduces t_r to 79 lines if `\looseness > 0`. The last line becomes “one hyphen! Oops!” and a following t_b adds 39 lines. It ends like the $t_b \sqcup t_r$ block in section 4’s second pass. We have $L_1(t_b \sqcup t_r, t_r) = 70$.

Let’s execute the paragraphs of section 5 using for `\looseness` the negative of the original result. Here it means that the paragraph should become longer as the original d is negative.

PiBiii: 12	PiBiiiPi: 13	$d_l = -1$
PiBviii: 32	PiBviiiPii: 36	$d_l = -4$
PiBxiii: 52	PiBxiiiPiii: 58	$d_l = -6$
PiBxviii: 72	PiBxviiiPiv: 80	$d_l = -8$
PiBxxiii: 92	PiBxxiiiPv: 102	$d_l = -10$
PiBxxviii: 112	PiBxxviiiPvi: 124	$d_l = -12$
PiBxxxiii: 132	PiBxxxiiiPvii: 146	$d_l = -14$
PiBxxxviii: 152	PiBxxxviiiPviii: 168	$d_l = -16$
PiBxliii: 172	PiBxliiiPix: 190	$d_l = -18$
PiBxlviii: 192	PiBxlviiiPx: 212	$d_l = -20$

This time the situation is different. `TeX` is not able to increase the first pass’ initial length by $l_i = -d$ except in the first case. Starting with the second line, `TeX` switches to the second pass and this increases the initial length compared to the result of the first pass by $-d$ as $n_2 = n_1 - d$. Now it is asked to increase this length by $l_i = -d$ to get $n_2 + l_i = n_1 + 2l_i$ lines. The paragraphs become longer by the value $2 \times \text{\code\looseness}$ compared to the left column.

The initial block t_i can be typeset in 14 lines instead of 13 and it keeps the same ending as in the first pass. Each t_r builds a rectangle of 22 lines instead of 20 if `\looseness > 2`.

9 Changing parameters of the second pass

What distinguishes the second from the first pass? `TeX` hyphenates words and accepts wider stretched glue [1, p. 96]. `TeX` allows us to switch these features on and off with two parameters: `\hyphenpenalty` and `\tolerance`, respectively. The assignment of the values 10000 and 100, resp., to these parameters limits the second pass to what the first can achieve. We get two intermediate states between the first and the second pass if one of these features is disabled.

Setting `\tolerance=100`. It should be no surprise that the text of section 4 gives the same results as shown in that section. We used the fact that only hyphenation is required for the second pass in section 6. This holds for the text of section 5 too.

That a zero `\parfillskip` does not change the data for these two texts should also be no surprise.

And the changed tolerance has no effect for the text of section 4 in the scenario of a nonzero `\looseness` in section 8 as everything is typeset in the first pass except the first line.

But a difference is seen for the text of section 5 and a positive `\looseness`. Here are the results:

PiBiii: 12	PiBiiiPi: 13	$d_l = -1$
PiBviii: 32	PiBviiiPii: 35	$d_l = -3$
PiBxiii: 52	PiBxiiiPiii: 57	$d_l = -5$
PiBxviii: 72	PiBxviiiPiv: 78	$d_l = -6$
PiBxxiii: 92	PiBxxiiiPv: 100	$d_l = -8$
PiBxxviii: 112	PiBxxviiiPvi: 122	$d_l = -10$
PiBxxxiii: 132	PiBxxxiiiPvii: 143	$d_l = -11$
PiBxxxviii: 152	PiBxxxviiiPviii: 165	$d_l = -13$
PiBxliii: 172	PiBxliiiPix: 187	$d_l = -15$
PiBxlviii: 192	PiBxlviiiPx: 208	$d_l = -16$

Except for the first line, `TeX` must switch to the second pass as before and starts therefore with a paragraph whose number of lines is larger, by the value of `\looseness` — as $l_i > 0$ — than in the first pass. But this time the reduced tolerance no longer allows `TeX` to make the paragraph a second time larger by l_i . We see smaller changes than before: $l_2 + 1 < 4 = 2l_2$, $l_3 + 2 < 2l_3$, $l_4 + 2 < 2l_4$, $l_5 + 3 < 2l_5$, etc. It is $l_i + 2j$ for $i \in \{3j, 3j + 1\}$ and $l_i + 2j + 1$ for $i = 3j + 2$ with $j = 0, 1, 2, \dots$. That is, the results establish a new infinite sequence. (The theorem can be applied to the subsequences with longer repetition blocks.)

No hyphenation. As hyphenation is important for the text of section 4, its disabling destroys any difference: The scenario of section 4 and the one of section 7 with the zero `\parfillskip` report in all cases $d = 0$. As the second pass is used only for the first case in section 8’s `\looseness` scenario with the text of section 4, only the first line can change. And it does: We get the value 40 for `\PiBiMii`.

For the text of section 5 we get new results. If we execute section 5’s paragraphs with the setting `\hyphenpenalty=10000` the numbers for d decrease and start now with 0 rather than -1 .

PiBiii: 12	PiiBiii: 12	$d = 0$
PiBviii: 32	PiiBviii: 33	$d = -1$
PiBxiii: 52	PiiBxiii: 54	$d = -2$
PiBxviii: 72	PiiBxviii: 75	$d = -3$
PiBxxiii: 92	PiiBxxiii: 96	$d = -4$
PiBxxviii: 112	PiiBxxviii: 117	$d = -5$
PiBxxxiii: 132	PiiBxxxiii: 138	$d = -6$
PiBxxxviii: 152	PiiBxxxviii: 159	$d = -7$
PiBxliii: 172	PiiBxliii: 180	$d = -8$
PiBxlviii: 192	PiiBxlviii: 201	$d = -9$

The same sequence for d is found if the text of section 5 is typeset with a zero `\parfillskip`. As

in the previous case, the values are changed by 1 compared to the ones shown in section 7.

The most interesting case is what happens with the text of section 5 and a nonzero `\looseness` if hyphenation is disabled. In the previous subsection we saw that a smaller tolerance can't achieve the results of section 8, where we get a paragraph in the second pass that increases by $2 \times \text{\code\looseness}$ lines; nevertheless, an infinite sequence appears.

PiBiii: 12	PiBiiiPi: 13	$d_l = -1$
PiBviii: 32	PiBviiiPii: 34	$d_l = -2$
PiBxiii: 52	PiBxiiiPiii: 56	$d_l = -4$
PiBxxviii: 72	PiBxxviiiPiv: 77	$d_l = -5$
PiBxxxiii: 92	PiBxxxiiiPv: 98	$d_l = -6$
PiBxxxviii: 112	PiBxxxviiiPvi: 119	$d_l = -7$
PiBxxxxiii: 132	PiBxxxxiiiPvii: 141	$d_l = -9$
PiBxxxxviii: 152	PiBxxxxviiiPviii: 162	$d_l = -10$
PiBxlxiii: 172	PiBxlxiiiPix: 183	$d_l = -11$
PiBxlxviii: 192	PiBxlxviiiPx: 204	$d_l = -12$

\TeX can't obey the request to extend the paragraph by `\looseness` lines except in the first line. But without hyphenation the paragraphs have more difficulty being extended after the switch to the second pass, i.e., the numbers in the second column are smaller than those in the previous subsection. That is, hyphenation is more important to get longer paragraphs than a larger tolerance in this case. But as lines 3 and 7 show there is sometimes an increase in the length of the paragraphs. This pattern builds an infinite sequence for $d_l: l_i + j + 1$ for $i = 4j + 3$ and $l_i + j$ for $i \in \{4j, 4j + 1, 4j + 2\}$; $j = 0, 1, \dots$

10 Final remarks and summary

The texts that I used in this article do not have any special features. I mean, there is no sequence of one letter words or every second space preceded by a punctuation mark, etc. We can safely assume that the results are valid for other paragraphs too. That text blocks are repeated instead of being written as “normal” text cannot be named a special feature of a particular paragraph.

Neither the `\hsize` nor the font `cmr9` plays an important rôle. Of course, a large `\hsize` makes it more difficult to change, say, the length between the two passes by two lines. A larger font size with wider characters makes it easier. For the chosen values — `\hsize = 225pt` and font `cmr9` — it seems that the length of the paragraphs for $d = \pm 2$ must have more than 30 lines. That is still a value that happens in practice. For $d = -3$ the text example needs 55 lines in the second pass so that a real text might achieve this. The length of section 4's paragraph with ≈ 120 lines for $d = 3$ hardly occurs in a document.

Section 4's answer for the speculation in the first section about how many lines we need for $d = 20$ turns out to be 1481. The value $d = -20$ is reached by the text of section 5 with 392 lines. Not surprising: It seems to be easier to produce longer paragraphs with the second pass than with the first.

It's possible to get a paragraph that becomes longer instead of getting shorter if we delete a single character in the text. The switch to the second pass can trigger this behavior. One example showed that the deletion of a character increases the number of lines by two while \TeX uses the first pass for both paragraphs. Thus, a switch between second and first pass is not necessary for this scenario. The results in section 6 show that paragraphs exist for which the deletion of a single character changes the number of lines by any given number.

After seeing the results of sections 4 and 5, one might expect that they can also be obtained with `\parfillskip=0pt`. Section 7's results prove this.

Section 7 also proves that the switch from \TeX 's default setting to `\parfillskip=0pt` might change the length of the paragraph. In [3] I showed that a reduction of two lines cannot be achieved with up to seven lines in the used test environment. In section 7 a text with 121 lines gives an upper bound for the required minimal length.

The reduction by `\looseness` works with section 4's paragraphs in the first pass. With those of section 5 we get an increase in the second pass. Therefore, we can find for every value of `\looseness` a paragraph that obeys it. See section 8.

The results of section 9 indicate — at least for one of the texts used in this article — that the ability to hyphenate words is more important for the difference of second and first pass than the higher value of `\tolerance` compared to `\pretolerance`.

References

- [1] Donald E. Knuth, *The \TeX book*, Volume A of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1984.
- [2] Udo Wermuth, “Tracing paragraphs,” *TUGboat* **37**:3 (2016), 358–373. Errata in *TUGboat* **38**:3 (2017), 414. tug.org/TUGboat/tb37-3/tb117wermuth.pdf
- [3] Udo Wermuth, “Can ‘`\parfillskip = 0pt`’ shorten a short paragraph in plain \TeX by two lines?,” *TUGboat* **43**:3 (2022), 343–350. tug.org/TUGboat/tb43-3/tb135wermuth-shorten.pdf

◇ Udo Wermuth
Dietzenbach, Germany
[u dot wermuth \(at\) icloud dot com](mailto:u_dot_wermuth(at)_icloud_dot_com)