ENHANCED PARAGRAPHING AND MICROTYPOGRAPHY FEATURES FOR N-T-ROFF

Roy Fisher

12/29/2017

CONTENTS

ILLUSTRATIONS

INTRODUCTION

This document describes a group of features designed to improve the uniformity of word spaces, the quality of line breaks and paragraph composition, and enhance the overall page appearance. There are also new methods and controls for letter adjustment ("microtypography"), and a mechanism for identifying troublesome areas that may benefit from individual attention. These features build on the improvements introduced in Heirloom Troff and brought forward in n-t-roff.

In paragraph adjust mode Heirloom uses a dynamic programming method similar to the one used by TEX. For each potential line break, characteristics that are important to good paragraph composition are graded and assessed a penalty for deviations from the desired value. After the entire paragraph has been analyzed, the combination of line breaks with the lowest total penalty is selected as the "least bad."

The word space, line breaking, and letter adjustment features are designed for paragraph adjust mode, but, like Heirloom, letter adjustment also works in single line mode. The features are implemented through a set of requests that provide controls for different aspects of paragraph composition. The controls can be adjusted to favor better line breaks, less hyphenation, more uniform word spaces, or balance them for a given text and page design. When troublesome paragraphs occur, the controls can be readjusted to alter the characteristics.

The new controls essentially provide a different way of using the existing code. The new code was added in a (mostly) plug-in fashion: n-t-roff was modified only as necessary to splice in the new controls. By default, the Heirloom/n-t-roff code runs and the new features are turned off. They are best used in extension level 3, and are activated by issuing requests that make use of them. They are available only to Troff, as Nroff has no typesetting capabilities.

Each of the paragraph characteristics is assigned a relative importance (penalty), and the characteristics compete against each other in a multi-way tug o' war. The most neutral balance (lowest total penalty) for the entire paragraph is deemed the "best" paragraph. The requests that control the characteristics can be grouped as follows:

WORD SPACE CONTROL

.wrdspc : Sets the range of preferred space sizes. A wider range allows more variation, tends to reduce hyphenation, and allows better line breaks at the cost of reduced word space consistency.

.wscalc : Determines the tolerance for loose or tight spaces relative to the desired size, based on the range established by **.wrdspc**.

LETTER ADJUSTMENT

.letcalc : Determines the way letter adjustment is applied, and whether it is distributed among the glyphs or among the glyphs and spaces.

.letthresh : Reserves a region around the desired space size in which no letter adjustment is applied, thereby reducing the overall amount of letter adjustment in the paragraph.

.letpen : Determines whether the effects of letter adjustment are taken into account when the line breaks are determined; can also apply a penalty to reduce excessive use.

.letstren : Modifies the rate of application. If the strength is zero, the line's natural proportions are taken into account.

.adjletpen : Disfavors adjacent lines if one uses a lot of letter stretch and the other uses a lot of letter shrink.

.letadj : The function of the space size threshold (the third argument) has been replaced by **.letthresh** when the new letter adjust methods are in use. Heirloom letter adjustment is not affected.

UNDESIRABLE CHARACTERISTICS

.adjpenalty : Disfavors consecutive lines when one is loose and the other tight.

.overrunpenalty : Progressively discourages last lines that are shorter than a user-specified length, and prevents lines that are too short.

DESIRABLE CHARACTERISTICS

.elppen : Favors or disfavors lines ending with a punctuation character. The companion request **.elpchar** defines a user specified list of characters; the characters are not limited to punctuation.

.lastlinestretch : Stretches the last line to full measure if its natural length falls within one en of the line length.

.linepenalty : Introduces a bias toward shorter paragraphs.

.looseness : Makes the paragraph shorter or longer. This is useful when eliminating a widow line or avoiding some other layout problem.

HYPHENATION

.exhyp : Discourages lines that end with a discretionary hyphen, as is used in hyphenated compound words.

.hypp : Hyphenation penalties are applied differently in order to obtain a better tradeoff between space size and end-of-line hyphenation. The behavior is more predictable and the penalty values have the same scale as all other penalties. Heirloom mode is not affected. A new argument $hypp_4$ discourages hyphenating the penultimate line of the paragraph.

CONSTRAINTS

.wsmin establishes a soft minimum word space size. It provides better resolution than the hard limit imposed by **.minss**.

MONITORING AND FEEDBACK

.wsmark : Places a number in the margin indicating the space size for each line. Useful for working with problem paragraphs.

.wswarn : Similar to **.wsmark**, but flags lines with space sizes smaller or larger than user-specified values, and optionally writes a message to `stderr`. It is useful for finding loose or tight lines over a large portion of a document.

The paragraph macro is a convenient place to define all of the normal settings and ensure they are always reset should they be changed to tune a paragraph. The following example lists the normal settings used for this document:

```
.do xflag 3
.padj
  . . .
.de pg
.   br
.   ft R
.   ps 11.6p
.   vs 15.5p
.   ti +15p
.   ss 12 0
.   minss 8
.   hypp 50 100 100 50
.   wrdspc 80 133
.   wscalc 6
.   wsmin 0
.   adjpenalty 50 80 133
.   overrunpenalty 10 25 16p
.   linepenalty 1
.   lastlinestretch 1
.   exhyp 50
.   elppen -3
.   letcalc 2
.   letadj 98 100 12 101.5 100
.   letthresh 86 125
.   letstren 100
.   letpen 8 1 1
.   adjletpen 2 100
..
```

Figure 1: Sample paragraph settings

The requests beginning with **.wrdspc** define the new methods and constraints, and the tradeoffs to be made among the paragraph's characteristics. The major characteristics are graphed in Figure 2 on page 7.

.do xflag 3 : Initiates extension level 3, enabling the long request names needed by these features. Paragraph-at-once mode, set by **.padj** or **.ad p**, is required for most of them.

.hypp 50 100 100 50 : The fourth argument discourages hyphenating the penultimate line of a paragraph. If the last word is hyphenated, the third penalty is applied instead. These penalty values disfavor hyphenating the last word more than the last line.

.wrdspc 80 133 : Defines the preferred space sizes to be from 80% to 133% of the desired size established by **.ss 12**. Spaces smaller than 80% or larger than 133% are discouraged at a higher marginal rate than spaces within the range. The minimum space size is set by **.minss**, the maximum is limited by the line length.

.wscalc 6 : Defines the shape of the curve that is used to grade the word spaces. This is a sixth-order curve, with a relatively flat base and steep skirts; it lightly penalizes spaces that are close to the desired size and sharply penalizes those beyond the values set by **.wrdspc**.

.wsmin 0 : Defines a soft minimum space size. The argument 0 switches the feature off; the minimum is enforced by **.minss**.

.adjpenalty 50 80 130 : Adjacent line incompatibility penalty; it applies a penalty of 50 to consecutive lines if one has spaces smaller than 80% and the other has spaces larger than 130%.

.overrunpenalty 10 25 16p : Discourages the last line of a paragraph from being shorter than 25% of full measure and prevents lines shorter than 16 points.

.linepenalty 1 : Slightly favors shorter paragraphs.

.lastlinestretch 1 : If the natural length of the last line falls within one en of full measure, the line is stretched to be fully justified.

.exhyp 50 : The penalty for breaking a line at an explicit hyphen is 50.

.elppen -3 : Lines that end with a punctuation character are penalized by −3 (a bonus of 3) to slightly encourage placing punctuation at the end of a line.

.letcalc 2 : Distributes letter adjustment among the glyphs; this method usually provides the best line breaks. The first page of the introduction and the first paragraphs of page 2 use **.letcalc 4**, which usually imparts a slightly more uniform overall appearance.

.letadj 98 100 12 101.5 100 : Enables dynamic letter spacing within a range of −2% to +1.5% of an en (−1% to +0.75% of an em). Glyph scaling is not applied. The third argument is not used with the new letter adjustment methods.

.letthresh 86 125 : Establishes a region between 86% and 125% of the nominal space size in which letter adjustment is not applied. If the space size crosses either threshold, letter adjustment is applied.

.letstren 100 : Letter adjustment is applied at the normal rate (100%).

.letpen 8 1 1 : Letter adjustment's effect on the space size is to be taken into account when determining line breaks, and a penalty of 8 is to be applied if the total amount of letter adjustment exceeds ±1%.

.adjletpen 2 100 : A small penalty of 2 is applied when consecutive lines have incompatible letter adjustment (they differ by more than ±100% of the amount specified by **.letpen**).

It isn't necessary to manually set all these values if you are using one of the presets (**.wscalc 0**, **10**, **11**, or **12**). The presets initialize some of the paragraph controls (the requests **.hypp** through **.elppen** in Figure 1), so it would only be necessary to make additional requests to configure the paragraph and letter adjust parameters that are not initialized by the preset, or to change a default setting.

The current values of the parameters are accessible through the read-only number registers listed on page 25.

For more information on the concepts of paragraph optimization and microtypography, see:

- Donald E. Knuth and Michael F. Plass, "Breaking Paragraphs Into Lines" [1];
- Hàn Thế Thành, "Micro-typography Extensions to the TEX Typesetting System" [2];
- Gunnar Ritter, "Justification in Heirloom Troff" [3].

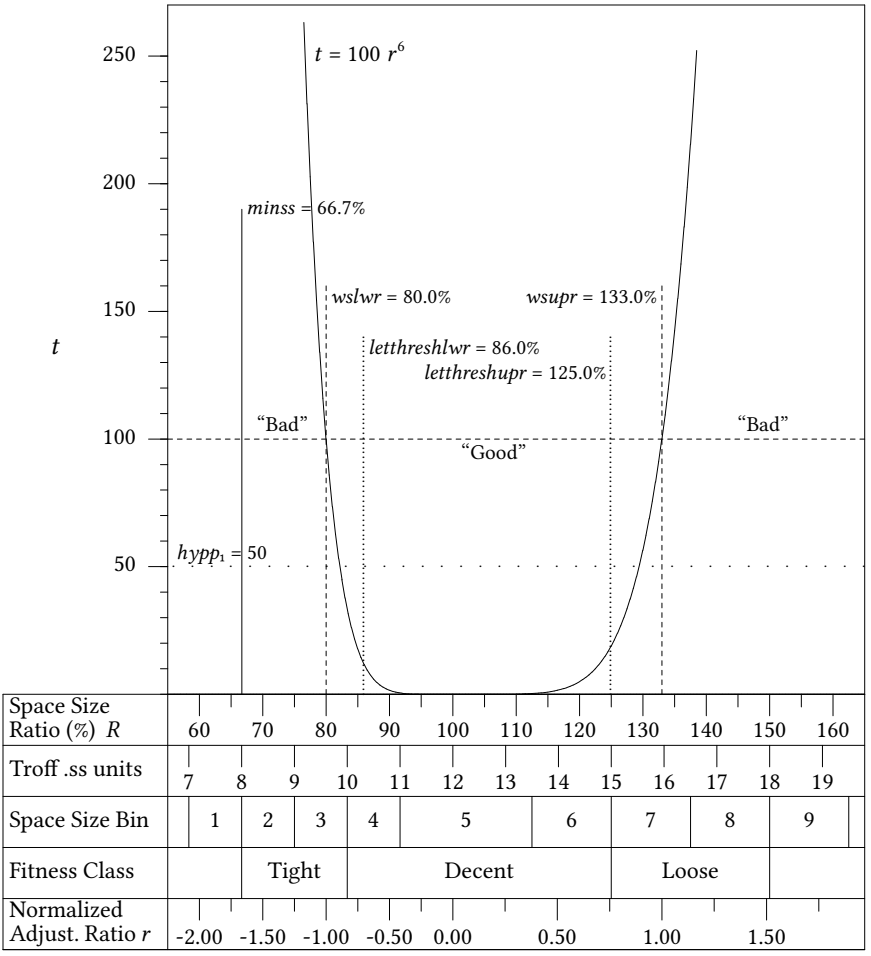

Figure 2: Schematic of the paragraph settings of Figure 1

The U-shaped curve represents the word space penalty *t* defined by **.wscalc 6** and **.wrdspc 80 133**. The range of preferred space sizes is 80% to 133% of the desired space size **.ss 12**, and the minimum space size is 66.7% (**.minss 8**). If the space size ratio *R* is between 86% and 125% (**.letthresh 86 125**), no letter adjustment is applied. Fitness classes are the standard TEX classes.

Penalties are added to the word space penalty curve, shifting it vertically by the amount of the penalty. If the line were hyphenated, the entire curve would be shifted upward by the value of $hypp_1$ (50).

.adjpenalty *pen thresh* (*threshupr*) *Default:* 0 50 0

Adjacent line incompatibility penalty. Discourages consecutive lines when one has loose word spaces and the other has tight spaces. The space size of each line is placed into a fitness class determined by the size of the word spaces and the value of *thresh*, and the penalty *pen* is applied if the fitness classes differ by more than one.

If *threshupr* is not specified, the value of *thresh* defines the class size in percent of the distance from the desired space size to the lower or upper bound for "good" sizes as defined by the **.wrdspc** request. For example, **.adjpenalty 100 50** establishes fitness classes that are 50% of the distance from the desired space size to the "good" boundaries and applies a penalty of 100 if two consecutive lines are more than one class apart. With the default **.wrdspc 66.7 150**, the "tight" class would be 83.3% and below, the "loose" class 125% and higher, and the "decent" class in between.

If *threshupr* is also specified, the meaning of the numbers changes. *thresh* becomes the threshold for shrunk spaces and *threshupr* becomes the threshold for stretched spaces, expressed in percent of the desired space size. This allows fixed shrink and stretch classes that are independent of **.wrdspc**. Setting **.adjpenalty 100 83.3 125** defines classes at the same points as above, but they do not scale with **.wrdspc**.

.elpchar *characters* *Default:* `.,;:?!'")]}`

Defines a custom list of line ending punctuation characters for use by **.elppen**. If no characters are specified, the default list is assigned. Characters can be appended to the current list by referencing the read-only register **\n[.elpchar]** followed by the new characters:

```
.elpchar \n[.elpchar]\(rq\[guillemotright]
```

adds the right quote and right guillemot: `.,;:!?'")]}”»` .

.elppen *pen* *Default:* 0

Line ending punctuation penalty. Applies penalty *pen* to lines that end with a punctuation character, excluding hyphens. *pen* > 0 discourages punctuation at the end of the line (for better appearance), and *pen* < 0 encourages it (for better readability). For values of *pen* < 0, best results are usually achieved with small values, −3 to −5. The default characters are from the ASCII set: `.,;:?!'")]}` . A custom list can be defined with **.elpchar**.

.exhyp *pen* *Default:* 0

Explicit hyphen penalty. Discourages line breaks at explicit hyphens, such as occur in hyphenated compound words. Explicit hyphens are not counted by **.hlm** or **.hypp** at present.

.hypp $hypp_1$ $hypp_2$ $hypp_3$ $hypp_4$ *Default:* 0 0 0 0

The new fourth argument, $hypp_4$, discourages hyphenating the penultimate line of a paragraph. If the last word is hyphenated, $hypp_3$ takes precedence if it is greater than zero. With the new **.wscalc** methods penalty values now have a more predictable behavior, although they will probably need to be considerably higher to have a similar effect.

.lastlinestretch 0 | 1 *Default:* 0

Stretches the last line of a paragraph to full measure if the line's natural length falls within one en of the line length. A 1 or no argument turns the feature on, 0 turns it off.

.linepenalty *pen* *Default:* 0

Favors paragraphs that have fewer lines. The penalty *pen* is applied to each line. A typical value is 1.0; useful values range from 0.25 to 10 or more. **.linepenalty** is functionally a type of overrun control.

.looseness *n* *Default:* 0

Alters the length of the paragraph by *n* lines. $n > 0$ increases the length, $n < 0$ decreases it. Turns itself off at the end of the paragraph. Sometimes it might be necessary to change the constraints, such as *minss*, letter adjustment, or tracking, to obtain the desired paragraph length.

This paragraph is one line longer than its natural length.

.overrunpenalty *pen thresh*$_1$ (*thresh*$_2$) *Default:* 0 25 0

Discourages overrun lines (short last lines). *thresh*$_1$ defines the length of the overrun line at which the penalty is first applied, expressed in percent of full measure; shorter lines are progressively penalized. The minimum penalty is *pen* at *thresh*$_1$; at *thresh*$_1$ / 2 the penalty is 2 × *pen*, at *thresh*$_1$ / 4 the penalty is 4 × *pen*, and so on. The last line may be lengthened, moved to the previous line, or left as-is, whichever gives the lowest total penalty. The optional argument *thresh*$_2$ defines a minimum length in units of distance (default: ems). If *thresh*$_1$ and *thresh*$_2$ are not specified, the current values are retained. Some care must be taken when choosing the progressive penalty: if it is set too high, the entire paragraph can be visibly degraded.

.overrunpenalty 10 25 progressively penalizes last lines shorter than 25% of full measure. At 25% the penalty is 10, at 12.5% it is 20, etc.

.overrunpenalty 1 0 30p prevents lines shorter than 30 points, but does not assess a progressive penalty. This is useful for ensuring that the last line is longer than the paragraph indent. For example, the paragraphs in this section have a first line indent of 15 points, so by setting *thresh*$_2$ ≥ 30p, last lines shorter than twice the paragraph indent, such as the "etc." in the previous paragraph, can be prevented should the progressive penalty allow them. The value of *pen* is not important in this case, but it must be greater than zero.

.overrunpenalty 10 25 2v applies a progressive penalty to lines that are shorter than 25% of full measure and prevents lines that are shorter than two times the baseline spacing.

.wrdspc *wslwr wsupr* *Default:* 66.7 150

Defines the preferred range of word spaces in percent of the desired space size. *wslwr* defines the lower end of the range, *wsupr* the upper. Values closer to 100 tend to reduce space size variation and increase hyphenation, values farther from 100 tend to do the opposite. Acceptable values are $0 \le wslwr \le 99$ and $101 \le wsupr \le 500$. The function of *wslwr* and *wsupr* is not to limit the size of the word spaces, but to define the space size adjustment ratios at which the word space penalty is 100, the dividing line between "good" to "bad."

In ragged (non justified) mode when paragraph adjustment is in effect, *wslwr* specifies the preferred minimum line length in percent of full measure; *wsupr* is irrelevant except that it must be at least 101. Thus, **.wrdspc 85 102** defines "good" lines to be at least 85% full; larger values favor lines that are more full, and smaller values allow more variation but may result in better line breaks.

.wscalc *n* *Default:* 0

Word space calculation method. Defines the shape of the curve used to grade the word spaces and the type of calculation used to apply the hyphenation penalties. Values of *n* are listed in Table 1.

The word space penalty curve defined by **.wscalc** and **.wrdspc** is the anchor for all other penalties. A penalty of zero is "perfect," 100 is the dividing line between "good" and "bad," and greater than 100 denotes "bad." **.wrdspc** defines the word space range at the "good"/"bad" line and **.wscalc** specifies the curve's shape. See Figures 4, 5, and 6 on pages 27–29 for graphs of the standard curve shapes.

Method 0 is the Heirloom Mode (default) preset. It runs the existing Heirloom code. It does not support **.wrdspc**, **.wsmin**, **.adjpenalty**, or the new letter adjustment features, but does support **.elppen**, **.exhyp**, **.lastlinestretch**, **.linepenalty**, **.looseness**, **.overrunpenalty**, **.wsmark**, and **.wswarn**. It sets all of the paragraph controls to zero in accordance with the Heirloom defaults [4], also sets the Heirloom letter adjustment method, **.letcalc 0**. The word space range is fixed at 0% to 160%.

Fig. 4, p. 27

Table 1 : Word space calculation methods

n	Type[a]	Description
0	Preset	Heirloom preset (default), runs the Heirloom code, has limited support of new features.
1	Curve	Same penalty calculation as **.wscalc 0**, but has full support of new features.
2–9	Curve	Continuous curves of order *n*.
10	Preset	TEX82-like configuration.
11	Preset	As described in the Knuth-Plass paper.
12	Preset	Adapted TEX82 curve. The default characteristics are the same as method 10, but penalties are applied like methods 2–9.
20–99	Curve	Two-phase curves. The tens digit defines the "good" portion of the curve, the ones digit defines the "bad" portion. Values are constrained to the range 2 through 9.

[a] A *curve* specifies only the word space penalty curve; a *preset* specifies the curve and various other controls.

Method 1 performs the Heirloom penalty calculation and supports all of the new features. The word space range can be adjusted with **.wrdspc**. **.wrdspc 1 160** approximates the Heirloom word space range.

Methods 2 through 9 apply a user specified curve of order *n*. The word space penalty *t* is calculated $t = 100 \mid r \mid^n$, where *r* is the normalized adjustment ratio.

Methods 10 and 11 apply a modified sixth-order curve based on the TEX calculations. Method 10 is based on TEX82 [5] and method 11 on the Knuth-Plass paper. These methods set **.wrdspc**, **.adjpenalty**, **.hypp**, and **.linepenalty** to the TEX defaults. The values for the current line hyphenation penalty and the line penalty have the TEX scale, and are not directly comparable to penalties used in methods 2 through 9. TEX's penalties and demerits must be converted as described on page 22. **.wscalc 10** can give a good approximation to TEX's paragraphing, but it is not intended to provide an exact match.

Fig. 5, p. 28

Method 12 is based on the TEX82 curve, but the line penalty and current line hyphenation penalty have been adapted to work the same as methods 2 through 9. Sets defaults for **.wrdspc**, **.adjpenalty**, **.hypp**, and **.linepenalty** to the TEX82 defaults.

Methods 20 through 99 apply a two-stage curve. The first digit (tens) defines the curve for the "good" spaces, the second digit (ones) for the "bad" spaces. **wscalc 36** applies a cubic curve $|r|^3$ to "good" space sizes and a sixth-order curve $|r|^6$ to "bad" ones, penalizing them at a much higher rate and making them even more undesirable. All curves are restricted to the range 2 through 9.

Fig. 6, p. 29

.wsmark 0 | 1 *Default:* 0

Places a number or character in the right margin to indicate the line's space size ratio; larger numbers indicate larger space sizes. A 1 or no argument switches it on, 0 switches it off. The margin character is the Bin Class as defined in Table 3 on page 26 and shown in Figure 4 on page 27. This feature is useful when fixing problem paragraphs.

Bin 5 is the central bin containing the nominal space size. Using the terminology for the standard TEX fitness classes, bins 4 through 6 (83.3–125%) are "decent," bins 2 and 3 (66.7–83.3%) are "tight," bins 7 and 8 (125–150%) are "loose," and spaces larger than bin 8 (150%) are "very loose." Spaces smaller than bin 2 (66.7%) are possibly too tight for running text, and those larger than bin 8 (150%) might be too loose, although they are sometimes unavoidable.

4 5 4 6 7 4 5 5 0 X

.wsmin *lwr* *Default:* 0

Defines a soft minimum space size in percent of the desired space size. The value of *lwr* must be larger than the hard limit set by **.minss** for there to be any effect. With **.ss 12** and **.minss 8**, **.wsmin 71** limits the minimum space size to 71% of the desired space size instead of the 66.7% imposed by **.minss**. But with **.minss 9**, **.wsmin 71** will have no effect because **.minss** keeps the minimum space size from being smaller than 75%.

.wswarn *level lwr upr* *Default:* 0 66.7 150

Similar to **.wsmark**, but only flags lines that may need attention. If a line has spaces smaller than *lwr* percent or larger than *upr* percent of the desired size, the line is flagged. If *level* ≥ 1, the word space bin is written in the margin next to the line; if *level* = 2, a message is also written to `stderr`. *level* = 0 turns the feature off. If *lwr* and *upr* are not supplied, the previous values are assumed.

With no arguments **.wswarn** is equivalent to **.wswarn 1**. The request **.wswarn 1 100 100** is equivalent to **.wsmark 1**.

.wswarn 2 75 137.5 flags only lines with spaces smaller than 75% or larger than 137.5% of nominal (Bins 2 and 8, the farther half of the "tight" and "loose" classes) and writes a message to `stderr`:

```
wswarn: wordspace bin 0 ratio 0.578482 on page 14
wswarn: wordspace bin X ratio 2.252395 on page 14
wswarn: wordspace bin B ratio 1.775976 on page 15
```

The corresponding lines are flagged by printing their bin class next to them in the margin. **.wswarn** is useful for checking a document or a large portion of one without having to examine it page by page.

If both **.wswarn** and **.wsmark** are in effect at the same time, **.wsmark** takes precedence for marking the lines in the margin, and the messages sent to `stderr` originate from **.wswarn**. B

.adjletpen *pen pct* *Default:* 0 100

Adjacent letter adjust penalty. For two consecutive lines, if the letter adjustment of one has a lot of shrink and the other has a lot of stretch, a penalty based on *pen* is applied if **.letpen**'s penalty is 2 or greater. If the letter adjustment of one line is shrunk by *pct* times **.letpen**'s argument *shrinkthresh*, and the other line is stretched by *pct* times **.letpen**'s argument *stretchthresh*, a penalty of *pen* is applied. When the amount of letter adjustment is greater than the threshold, the penalty increases sharply. If **.letpen**'s penalty is less than 2, **.adjletpen** has no effect.

Given **.adjletpen 30 100** and **.letpen 10 0.5 0.65**, a penalty of 30 is applied if the letter adjustment of one line is shrunk by 100% × 0.5% (0.5%) and the other is stretched by 100% × 0.65% (0.65%).

.letcalc *n* *Default:* 0

Defines the method used to calculate letter adjustment. The values of *n* are:

Table 2 : Letter adjustment methods

n	Description
0	Heirloom mode (default)
1	Letter adjustment last (near-bad lines only)
2	Letter adjustment first, then word space adjustment
3	Letter adjustment proportional to space adjustment
4	Letter adjustment distributed among glyphs and spaces

Method 0 executes the Heirloom code. Letter shrink is applied when glyphs would not otherwise fit on the line, and stretch is applied when the spaces are larger than the letter stretch threshold defined by the third argument to **.letadj**. Letter adjustment is distributed among the glyphs and spaces. Method 0 does not support any of the new features.

Method 1 is loosely based on the Heirloom approach. No letter adjustment is applied until the space size approaches the values set by the **.wrdspc** request. For example, with **.wrdspc 80 133**, letter adjustment is not applied until the spaces become smaller than 81% or larger than 132%. Letter adjustment is then done per method **.letcalc 2**. This method reduces overall glyph distortion but improves line breaks and word space uniformity less than the **.letcalc 2** defaults. **.letcalc 1** is equivalent to **.letcalc 2** combined with **.letthresh *wslwr*+1 *wsupr*–1**.

Method 2 applies letter adjustment before changing the size of the word spaces. This method has the best word space uniformity and line break quality, but the most glyph distortion.

Method 3 applies letter adjustment based on the amount of word space adjustment. Word spaces near the desired size have little letter adjustment, borderline "bad" spaces have the maximum.

Method 4 distributes letter adjustment among all the glyphs and spaces. Its line breaking characteristics are similar to Method 2, but the overall amount of letter adjustment is less and the line breaks are sometimes not quite as good.

Methods 1, 2, 3, and 4 apply dynamic letter spacing and glyph scaling simultaneously and in proportion to the values defined by **.letadj**. Method 0 applies letter spacing and glyph scaling in equal amounts until one is used up, then applies it from whichever remains.

Methods 2, 3, and 4 are compatible with all letter adjustment requests. Method 1's settings cannot be independently changed.

As with the Heirloom code, letter adjustment is switched on or off with the **.letadj** request. **.letadj *lspmin lshmin letss lspmax lshmax*** turns letter adjustment on when *letss* > 0, and **.letadj** with no arguments turns it off. With the new methods the value of *letss* is irrelevant, except that it must be greater than zero to switch letter adjustment on.

If letter adjustment is switched off by setting the third argument to zero, e.g., **.letadj *lspmin lshmin* 0**, some tight lines that have a space size near *minss* will not justify correctly if *lspmin* and *lshmin* are not 100. The underlying Heirloom code always takes all available letter shrink into account when determining whether the text will fit on the line, but the new methods will not apply it if *letss* is zero.

.letpen *pen shrinkthresh stretchthresh* *Default:* 0 1.0 1.0

Defines whether the effect of letter adjustment on space size is taken into account when determining line breaks.

If *pen* = 0, the effects of letter adjustment are not taken into account but letter adjustment is applied when the lines are output.

If *pen* = 1, the effects of letter adjustment are taken into account. Letter adjustment is used freely within the limits established by **.letadj**.

If *pen* ≥ 2, the effects of letter adjustment are taken into account and a penalty is applied to discourage excessive use. The penalty has a reference value of *pen* and varies based on the threshold values *shrinkthresh* and *stretchthresh*, similar to the word space penalty. **.letpen 10 0.5 0.65** applies a penalty of 10 when the total amount of letter adjustment is shrunk by 0.5% or stretched by 0.65%. The penalty is progressive: it is zero when no letter adjustment is applied, and it becomes sharply greater than *pen* when the amount of letter adjustment is greater than the threshold values.

.letshp *lwr upr* *Default:* x x

Changes the dynamic glyph scaling (letter shaping) limits to *lwr* and *upr*. Equivalent to the request **.letadj x *lwr* x x *upr***, but is less error prone and does not throw a "bad number" message. Useful when tweaking paragraphs. **.letshp 99.6 101** shrinks the glyphs by a maximum of 0.4% or stretches them by a maximum of 1.0% of their natural width.

.letspc *lwr upr* *Default:* x x

Changes the dynamic letter spacing limits to *lwr* and *upr*. Equivalent to the request **.letadj *letlwr* x x *letupr* x**, but is less error prone and does not throw a "bad number" message because of the *x* placeholders. Useful when tweaking paragraphs.

Like Heirloom's **.letadj** request, the amount of letter spacing is one half of the amount we would expect from the numbers because the basis is the en instead of the em. With **.letspc 99.6 101**, the interletter

spacing is reduced by a maximum of 0.4% of an en (0.2% of an em) or increased by a maximum of 1% of an en (0.5% of an em).

.letstren *strength* *Default:* 100

Changes the rate at which letter adjustment is applied, expressed in percent; the normal rate is 100%. If *strength* is 90, letter adjustment is applied at the rate of 90%; *strength* > 100 increases the rate. Setting *strength* to zero is a special case; it will take into account the line's inherent proportion of total character length to total space length until all letter adjustment has been used up. The maximum amount of letter adjustment is not affected by **.letstren**: it is still limited to the values specified by **.letadj**.

.letthresh *letthreshlwr letthreshupr* *Default:* 100 100

Establishes a region surrounding the desired space size where no letter adjustment is applied. The values are expressed in percent of the desired space size. *letthreshlwr* establishes the smallest space size in this region, *letthreshupr* the largest. The value of *letthreshlwr* must be 100 or less, and *letthreshupr* must be 100 or greater. With the new **.letcalc** methods, *letthreshupr* replaces the function of *letss* (the third argument to the **.letadj** request).

.letthresh 83 125 applies no letter adjustment if the word spaces for the line are between 83% and 125% of the desired size. Within this region only the space size is allowed to change.

.letthresh 100 100 sets the size of the region to zero and applies letter adjustment to all lines.

Letter adjustment requests **.letadj**, **.letcalc**, **letstren**, **.letthresh**, **.letshp**, and **.letspc** can also be applied in single-line mode. They can markedly improve line breaks and overall appearance even without the use of paragraph-at-once justification.

PARAGRAPH CONTROLS

- The Heirloom penalty curves, **.wscalc 0** and **1**, have a strong interaction between the word space ratio and the hyphenation penalties. To obtain an Heirloom-like response without this interaction, set:

  ```
  .wscalc 3
  .wrdspc 0 160
  ```

  The Heirloom curve shape will shrink the space size a great deal to avoid a penalty item, so it is essential to control the minimum space size with **.minss**, which acts as a brick wall surrogate for *wslwr*. With **.wscalc 2** and up, hyphenation penalties will need to be considerably higher (two or three times as much) than with the Heirloom curves for reasonably similar results. Due to Heirloom's interaction with word spaces, a direct conversion is not possible.

- To obtain settings similar to the TEX82 defaults:

  ```
  .wscalc 10
  .ss 12
  .minss 8
  .hy 4
  .hlm 0
  ```

  .wscalc 10 sets several paragraph parameters to the TEX82 defaults. All other paragraphing parameters are set to zero.

  ```
  .wrdspc 66.7 150
  .hypp 50 100 0 50
  .exhyp 25
  .adjpenalty 100 50
  .linepenalty 10
  ```

To convert penalties and demerits from TEX's units, in general:

- Square penalties then divide by 100
- Divide demerits by 100

With **.wscalc 10** and **.wscalc 11**, some conversions need special attention because of the way an item is used:

.wscalc 10	TEX Conversion	Type
`.exhyp` *p*	`\explicithyphenpenalty`^{2} / 100	Normal
`.hypp` p_1 `x x x`	`\hyphenpenalty`	Special
`.hypp x` p_2 `x x`	`\doublehyphendemerits` / 100	Normal
`.hypp x x x` p_4	`\finalhyphendemerits` / 100	Normal
`.adjpenalty` *p*	`\adjdemerits` / 100	Normal
`.linepenalty` *p*	`\linepenalty`	Special

The current line hyphenation penalty, $hypp_1$, and the line penalty are squared before they are applied. The last word hyphenation penalty, $hypp_3$, is not supported by TEX.

- **.wscalc 11**, the preset based on the Knuth-Plass paper, is similar to **.wscalc 10**, but the current line's hyphenation penalty interacts with both the space size and the line penalty. The defaults are the same as **.wscalc 10** except:

```
.hypp 50 30 0 50
.linepenalty 1
```

- **.wscalc 12**, the adapted TEX82 preset, has the default behavior of **.wscalc 10**, but *linepenalty* does not interact with the word space penalty and $hypp_1$ is not squared before it is applied. The defaults are the same as **.wscalc 10** except:

```
.hypp 25 100 0 50
.linepenalty 1
```

LETTER ADJUSTMENT

- To adapt an Heirloom configuration to use the new controls:
  - Set *letthreshlwr* slightly larger than the ratio of *minss* / *ss*. If *minss* = 9 and *ss* = 12, set *letthreshlwr* to about 76% or 77%.
  - Set *letthreshupr* to *letss* / *ss*. For *letss* = 16 and *ss* = 12, that is 133%. (*letss*, the letter adjustment space size threshold, is the third argument to **.letadj**.)
  - Use **.letcalc 4**, to distribute letter adjustment among glyphs and spaces (similar to Heirloom's method), or **.letcalc 2**.
  - Heirloom does not take the effects of letter adjustment into account when it determines line breaks, other than to know whether the text will fit on the line. To match this behavior use **.letpen 0**. To account for the effects of letter adjustment use **.letpen 1**.
  - **.wscalc 1** replicates the Heirloom word space penalty curve and supports all of the new controls, or use **.wscalc 3** as described previously. Set the wordspace range as desired.

  The corresponding requests are:

  ```
  .letthresh 77 133
  .letcalc 4
  .letpen 0
  .wscalc 1
  .wrdspc 0 160
  .minss 9
  ```

  The resulting paragraph will be similar to the Heirloom defaults (**.letcalc 0** and **.wscalc 0**), but will probably not be identical.

- To set letter adjustment to values similar to those described in Hàn Thế Thành's pdfTeX thesis:

```
.letcalc 2
.letadj 100 98 12 100 102
.letpen 1
.letthresh 100 100
.letstren 100
```

READ-ONLY NUMBER REGISTERS

f = floating point, i = integer, s = string, % = percent, u = units

Paragraphing

.adjpenalty	f		adjacent line incompatibility penalty
.adjthreshold	f	%	adjacent line incompatibility threshold
.adjthreshupr	f	%	adjacent line incompatibility upper thresh.
.elpchar	s		end of line punctuation penalty characters
.elppen	f		end of line punctuation penalty
.exhyp	f		explicit hyphenation penalty
.hypp4	f		penultimate line hyphenation penalty
.linepenalty	f		penalty for each line
.looseness	i		looseness lines
.overrunmin	i	u	last line minimum length
.overrunpenalty	f		progressive overrun penalty
.overrunthreshold	f	%	threshold for progressive overrun penalty
.wscalc	i		word space calculation method
.wslwr	f	%	lower “good” word space boundary
.wsmark	i		word space mark level
.wsmin	f	%	minimum word space
.wsupr	f	%	upper “good” word space boundary
.wswarn	i		word space warn level
.wswarnlwr	f	%	word space warn lower threshold
.wswarnupr	f	%	word space warn upper threshold

Letter Adjustment

.adjlapenalty	f		adjacent line letter adjust penalty
.adjlathreshold	f		adjacent line threshold multiplier
.letcalc	i		letter adjustment method
.letpen	i		letter adjustment penalty
.letpenlwr	f	%	letter adjustment penalty lower reference
.letpenupr	f	%	letter adjustment penalty upper reference
.letstren	f	%	letter adjustment strength multiplier
.letthreshlwr	f	%	exclusion zone lower threshold
.letthreshupr	f	%	exclusion zone upper threshold

Figure 3

Table 3: Bin classes and space adjustment ratios

Bin Class[a]	VFC Class[b]	TEX82 Fitness Class[c]	TEX Class Description	Raw Adjustment Ratio, *R* Lower	Raw Adjustment Ratio, *R* Upper	Normalized[d] Adjustment Ratio, *r* Lower	Normalized[d] Adjustment Ratio, *r* Upper
f	−5	-	-	0	0.0833	−3.00	−2.75
e	−5	-	-	0.0833	0.1667	−2.75	−2.50
d	−4	-	-	0.1667	0.2500	−2.50	−2.25
c	−4	-	-	0.2500	0.3333	−2.25	−2.00
b	−3	-	-	0.3333	0.4167	−2.00	−1.75
a	−3	-	-	0.4167	0.5000	−1.75	−1.50
0	−2	-	-	0.5000	0.5833	−1.50	−1.25
1	−2	-	-	0.5833	0.6666	−1.25	−1.00
2	−1	3	Tight	0.6666	0.7500	−1.00	−0.75
3	−1	3	Tight	0.7500	0.8333	−0.75	−0.50
4	0	2	Decent	0.8333	0.9167	−0.50	−0.25
5	0	2	Decent	0.9167	1.1250	−0.25	0.25
6	0	2	Decent	1.1250	1.2500	0.25	0.50
7	1	1	Loose	1.2500	1.3750	0.50	0.75
8	1	1	Loose	1.3750	1.5000	0.75	1.00
9	2	0	Very loose	1.5000	1.6250	1.00	1.25
A	2	0	‘‘	1.6250	1.7500	1.25	1.50
B	3	0	‘‘	1.7500	1.8750	1.50	1.75
C	3	0	‘‘	1.8750	2.0000	1.75	2.00
X	4	0	Awful[e]	> 2.0000	-	> 2.00	-

Notes:

[a] The Bin Classes are printed in the margin by the **.wsmark** and **.wswarn** requests. The relationships of the Bin Classes, TEX Fitness Classes, TEX Class Descriptions, and Raw Adjustment Ratios are always as shown. Bin Class 5 contains the nominal space size.

[b] Variable Fitness Classes are used internally for applying the adjacent line incompatibility penalty. If consecutive lines are farther apart than one VFC, a penalty is applied. Depending on the way the **.adjpenalty** request is specified, the classes can scale to track the word space range or remain fixed and independent of it.

[c] The TEX82 fitness classes always have the same relationship to the Raw Adjustment Ratios. All spaces with raw adjustment ratios *R* greater than 1.5 are assigned to Class 0. TEX does not support space sizes smaller than two-thirds of the nominal size.

[d] The Normalized Adjustment Ratios *r* shown here are the defaults, and correspond to the TEX range of word spaces: 66.7% to 150% of the desired size (**.wrdspc 66.7 150**). The Normalized Adjustment Ratio *r* is scaled to the **.wrdspc** range so that when the Raw Adjustment Ratio $R = 0.6667$, $r = -1.0$, and when $R = 1.50$, $r = +1.0$. Changing the **.wrdspc** range causes *r* to change accordingly.

[e] “Awful” is not an actual fitness class, but a description borrowed from *The TEXbook* [6]. It is not the same as “awful bad,” which is orders of magnitude worse.

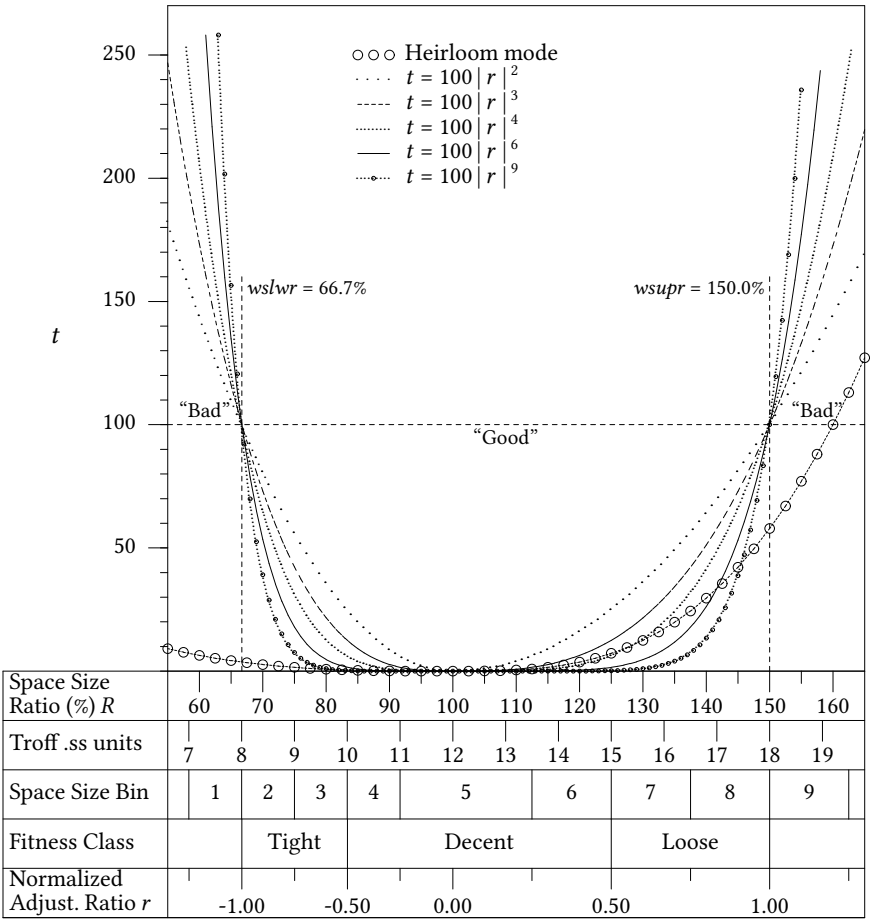

Figure 4: Word space penalty curves

Word space penalty *t* graphed as a function of the normalized space size adjustment ratio *r* for **.wscalc** curves 2, 3, 4, 6, and 9; the desired space size is **.ss 12** and the preferred word space range is the default, **.wrdspc 66.7 150**, except for the Heirloom curve, which has an effective range of 0% to 160%.

The table shows the relationships of the raw space size adjustment ratio *R*, Troff space size units, space size bins, TEX's standard fitness classes, and the normalized adjustment ratio *r*. The normalized adjustment ratio is determined by scaling the raw adjustment ratio *R* to the *wslwr* and *wsupr* arguments of **.wrdspc**; the corresponding penalty is $t = 100 \mid r \mid^n$. The space size bins and TEX fitness classes are always related to the raw adjustment ratio *R* as shown.

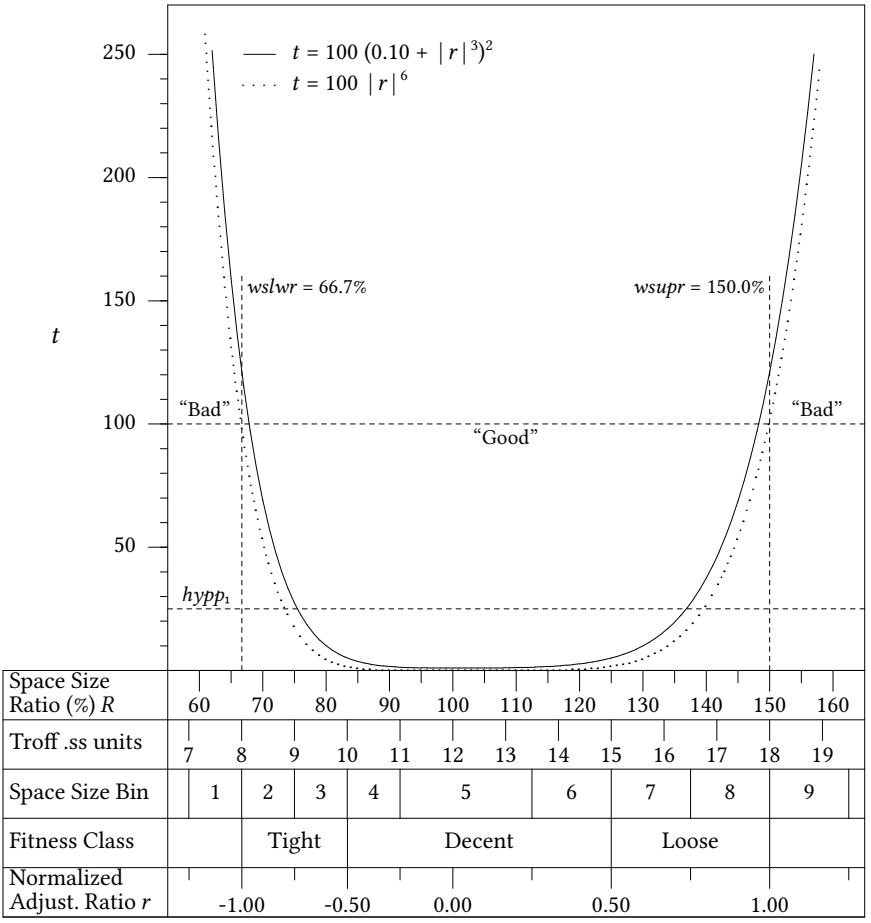

Figure 5: Default word space penalty curve for .wscalc 10 and .wscalc 12

The default word space penalty curve for **.wscalc 10** and **.wscalc 12** compared to an unmodified sixth-order curve. The TEX82-like curve has greater selectivity within the "good" range and a slightly narrowed "good" word space range. The default values of *wslwr*, *wsupr*, and $hypp_1$ are shown.

With **.wscalc 10** the default line penalty, 10 (0.10), shifts the curve vertically by 1 and changes the curvature and the effective values for *wslwr* and *wsupr*. Increasing the line penalty increases the curvature and narrows the word space range. When the line penalty is zero the curves are identical.

With **.wscalc 12** a fixed value of 0.10 is built into the curve shape. The line penalty is added to *t* and shifts the curve vertically without altering its shape.

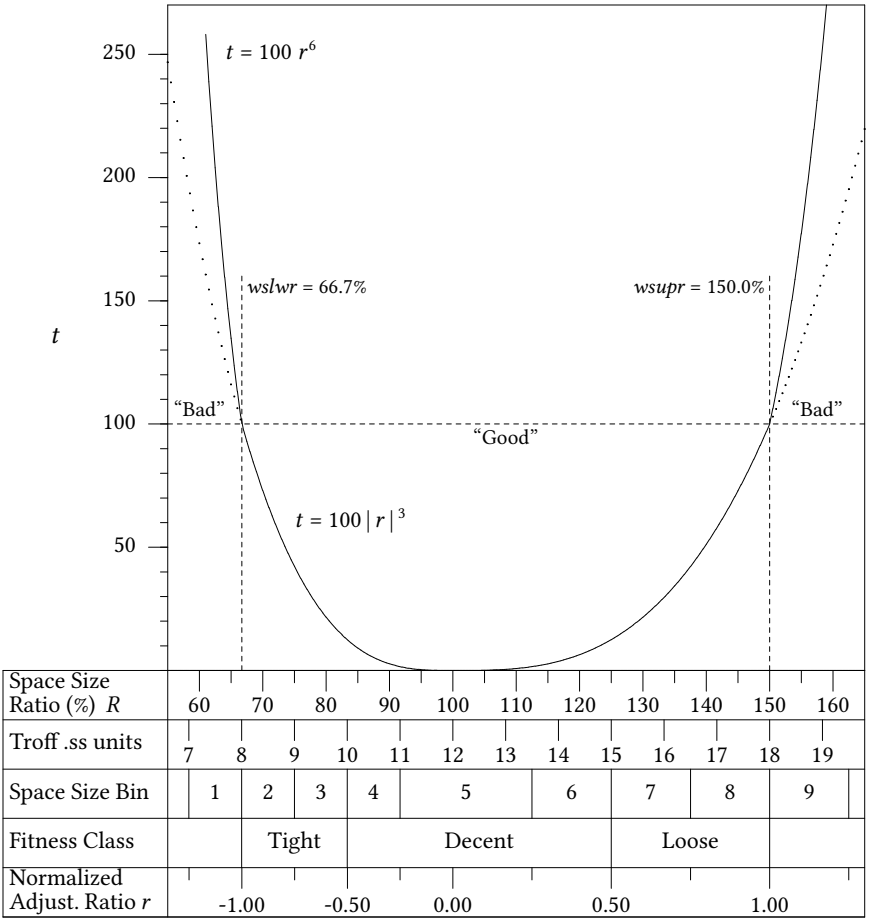

Figure 6: Two-stage penalty curve **.wscalc 36**

Two-stage curves apply the first digit (tens) to the "good" portion of the penalty curve and the second digit (ones) to the "bad" portion. With **.wscalc 36**, "good" spaces will be penalized by $t = 100 \mid r \mid^3$ and "bad" spaces by $t = 100\ r^6$, causing them to be disfavored at a greater rate. The dotted line represents the unused portion of of the cubic curve. *wslwr* and *wsupr* are the global defaults.

References:

[1] Donald E. Knuth and Michael F. Plass, “Breaking Paragraphs Into Lines.” *Software—Practice and Experience*, Vol. 11, 1981, 1119–1184.

[2] Hàn Thế Thành, “Micro-typography Extensions to the TeX Typesetting System,” Ph.D. dissertation, Faculty of Informatics, Masaryk University Brno, 2000.

[3] Gunnar Ritter, “Justification in Heirloom Troff.” Nov. 11, 2006. PDF. http://heirloom.sourceforge.net/doctools/just.pdf

[4] Joseph F. Osanna, Brian W. Kernighan, Gunnar Ritter, *et al.*, “Nroff/Troff User’s Manual,” Feb 10, 2016. PDF. http://n-t-roff.github.io/heirloom/doctools/troff.pdf.

[5] Donald E. Knuth, “TEX82,” Annotated program listing, Feb. 3, 2014. PDF. http://www.tug.org/texlive//devsrc/Master/texmf-dist/doc/generic/knuth/tex/tex.pdf

[6] Donald E. Knuth, *The TEXbook*, Boston, Mass.: Addison-Wesley Publishing Company, 1984.

The text is Libertinus Serif, an updated fork of Linux Libertine, set 11.6/15.5 with a measure of 27 picas. The fixed width font is Computer Modern Typewriter. All graphics and inserts are “live” using the grap, pic, tbl, and eqn preprocessors.