PARAGRAPHING AND MICROTYPOGRAPHY FEATURES

A QUICK REFERENCE

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 made in Heirloom *troff* and carried 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 topics covered in this document include: a sample paragraph macro, a one-page summary of the requests, detailed descriptions of the paragraph and letter adjustment requests, examples of the features in use, and illustrations that show the theory of operation, effects of the penalties, and relationships of the space size measurements.

Initially, the Heirloom code runs and the new features are turned off. They become available in extension levels 2 and 3 (**.do xflag 3** is recommended), and are turned on by issuing requests that make use of them, for example, **.wscalc** and **.letcalc**. The adjustment values are typically expressed in percent of the nominal space size or character size, or as a penalty number, with zero being no penalty and 100 the dividing line between "good" and "bad." The paragraph macro is a good place to define the document's normal settings:

```
.de pg
. br
. ft R
. ps 12
. vs 16p
. ti +12p
. ad pb
. ss 12 0
. minss 8
. wscalc 4
. wrdspc 75 140
. wsmin 0
. adjpenalty 100 80 130
. overrunpenalty 40 25 18p
. lastlinestretch 1
. linepenalty 1
. exhyp 25
. elppen -3
. letadj 99.5 99.5 12 101 100.5
. letcalc 2
. letthresh 100 100
. letstren 100
. letpen 5 0.6 0.7
. adjletpen 2 100
.
. \" other paragraph settings, etc.
.
..
```

SUMMARY OF REQUESTS

Paragraph Control

.wscalc	Defines the curve that word spaces are graded on.
.wrdspc	Defines the range of "good" word spaces.
.adjpenalty	Disfavors adjacent lines that are "loose" and "tight."
.lastlinestretch	Justifies a last line that is nearly full measure.
.linepenalty	Favors shorter paragraphs, discourages overrun lines.
.looseness	Makes paragraphs longer or shorter.
.overrunpenalty	Discourages short last lines.
.elppen	Disfavors or favors lines ending with a punctuation character.
.elpchar	Custom user-defined list of punctuation characters for **.elppen**.
.exhyp	Penalizes lines that end with an explicit hyphen.
.wsmin	Defines a soft minimum word space.
.wsmark	Marks word spaces in the margin.
.wswarn	Marks bad word spaces in the margin and optionally sends a message to `stderr`.

Letter Adjustment

.letcalc	Defines the type of letter adjustment.
.letpen	Includes effects of letter adjustment when determining line breaks.
.adjletpen	Disfavors adjacent lines with incompatible letter adjustment.
.letshp	Overrides glyph scaling settings.
.letspc	Overrides dynamic letter spacing settings.
.letstren	Letter adjustment strength.
.letthresh	Defines a range of space sizes to which no letter adjustment is applied.

.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. The penalty curves defined by *n* are:

n	Penalty Curve	Description
0	$t = (r + hyp)^3 + p$	Heirloom mode
1	$t = (r + hyp)^3 + p$	Heirloom penalty calculation
2	$t = r^2 + hyp + p$	Quadratic
3	$t = r^3 + hyp + p$	Cubic
4	$t = r^4 + hyp + p$	Quartic
5	$t = r^6 + hyp + p$	TEX compatibility mode
6	$t = r^6 + hyp + p$	TEX penalty calculation
7	$t = (1 + r^3 + hyp_1)^2 + p$	Knuth-Plass paper (1981)

Where:

t	is the total penalty for the line;
r	is the absolute value of the normalized space adjustment ratio;
hyp	is the sum of the hyphenation penalties;
hyp_1	is the hyphenation penalty for the current line only;
p	is the sum of all other penalties.

Mode 0 executes the existing Heirloom code. It does not support the features **.wrdspc**, **.wsmin**, or **.adjpenalty**, or any of the new letter adjustment features. It does support **.elppen**, **.exhyp**, **.lastlinestretch**, **.linepenalty**, **.looseness**, **.overrunpenalty**, **.wsmark**, and **.wswarn**.

Mode 1 performs the Heirloom penalty calculation and supports all of the new features.

For curves 2, 3, and 4, a second digit may be appended to *n* to invoke a steeper rejection curve for lines that have "bad" word spaces. Thus, **.wscalc 24** defines a quadratic penalty curve r^2 if the word spaces are

"good" and a quartic curve r^4 if the word spaces are "bad." A custom "bad" curve must be steeper than the "good" curve: $n = 45$ is valid, but $n = 42$ reverts to $n = 4$ (44). Invalid choices default to curve 6.

Mode 5, in addition to using the TEX penalty curve, also presets a number of features to values that are comparable to the TEX defaults: **.wrdspc 66.7 150**, **.adjpenalty 100 50**, **.hypp 25 100 50**, **.linepenalty 1**, and **.lastlinestretch 1**.

Mode 6 applies the TEX penalty calculation, but does not preset any features. With modes 5 and 6, the TEX82 calculation has been slightly altered to use floating point numbers and apply penalties consistently.

For modes 0 and 1, the hyphenation penalties defined by **.hypp** retain the same strength as in Heirloom: the effective value is the input hyphenation penalty divided by 50. For the new methods 2 through 7, the effective value is the input value divided by 100; this keeps all of the penalties for the various requests at the same strength for the same user inputs. Inline penalties **\j** and **\J** retain the Heirloom behavior.

.wrdspc *lwr upr* *Default:* 66.7 150

Defines the preferred range of word spaces in percent of the desired space size. *lwr* defines the smallest, *upr* the largest. Valid ranges are: $1.0 \le lwr \le 99$ and $101 \le upr \le 500$. Values closer to 100 tend to reduce word space variation and increase hyphenation, values farther from 100 tend to do the opposite. Default values are $lwr = 66.7$ and $upr = 150$. The values of *lwr* and *upr* do not limit the size of the word spaces, but define the space size at which the meaning of the word space penalty changes from "good" to "bad" and increases more rapidly (see Figure 1). In Heirloom mode, the values are preassigned and cannot be changed; they are equivalent to $lwr = 0$ and $upr = 160$.

In ragged (non justified) mode, when paragraph adjustment is in effect, *lwr* specifies the preferred minimum line length in percent of full measure; *upr* is irrelevant except that it must be a valid input. Thus, **.wrdspc 85 102** defines "good" lines to be at least 85% full. Changing this value changes the appearance of the paragraph: larger values will favor lines that are more full, smaller values allow more variation.

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

Adjacent line incompatibility penalty. Discourages consecutive lines with "loose" and "tight" word spaces. Fitness classes are established based on the value of *thresh*, and the word space of each line is placed into the corresponding fitness class. The penalty *pen* is applied if two lines are farther apart than one fitness class. The central class, termed "decent," is larger than the others: it is the width of one "tight" class plus one "loose" class. 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 specified by the **.wrdspc** request (see Figure 1).

.adjpenalty 100 50 establishes fitness classes that are 50% of the distance from the desired space size to the "good" boundary, and applies a penalty of 100 if two consecutive lines are more than one class apart (Figure 2). The classes scale with the values of **.wrdspc** (Figure 4).

If *threshupr* is also specified, the meaning of the numbers changes. *thresh* becomes the lower threshold (for shrunk spaces) and *threshupr* becomes the upper threshold (for stretched spaces), expressed in percent of the desired space size. This allows fixed shrink and stretch classes that are independent of **.wrdspc** values. If we have requested **.adjpenalty 100 83.3 125**, the "decent" class ranges from 83.3% to 125% of the nominal space size; these classes are the same as those produced by **.adjpenalty 100 50** and **.wrdspc 66.7 150**, but are *not* the same as those of **.adjpenalty 100 50** and **.wrdspc 75 140** (87.5% to 120%).

For *wscalc* modes 1, 2, 3, 4, 6, and 7, a "decent" line is not penalized; in TEX mode (5), a "decent" line is penalized if it is more than one class from the preceding line. Heirloom mode (0) is not supported.

.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 character list can be defined with **.elpchar** to add curly quotes, guillemots, etc.

.elpchar *character list* *Default:* . , ; : ? ! ’ ")] }

Defines a custom list of line ending punctuation characters. If no list is specified, the default list is assigned. Characters can be appended to the default list with **.elpchar \n[.elpchar]** followed by the new characters. For example, **.elpchar \n[.elpchar]\(rq\[guillemotright]** changes .,;:!?"")]} to .,;:!?"")]}”», and **.elpchar** changes it back to .,;:!?"")]}.

.exhyp *pen* *Default:* 0

Explicit hyphen penalty. Discourages line breaks at explicit hyphens, such as occur in hyphenated compound words.

.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. This feature was borrowed from TEX, but it does not behave quite the same due to implementation differences.

.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. In some cases it might be necessary to adjust the constraints, such as *minss*.

.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* / 2 at *thresh*$_1$; at *thresh*$_1$ / 2 the penalty is *pen*; at *thresh*$_1$ / 4 the penalty is 2 × *pen* ; and so on. The optional argument *thresh*$_2$ establishes a minimum length for the last line in horizontal units of measure (default: ems). Shorter lines are heavily penalized. If *thresh*$_1$ and *thresh*$_2$ are not specified, the current values are retained.

.overrunpenalty 40 25 progressively penalizes last lines shorter than 25% of full measure. At 25% the penalty is 20; at 12.5% it is 40; at 6.25% it is 80, etc.

.overrunpenalty 40 25 16p applies a progressive penalty to lines that are shorter than 25% of full measure, and attempts to prevent lines that are shorter than 16 points. This is useful when we want to make sure that the last line is longer than the paragraph indent. For example, the paragraphs in this section have a first line indent of 12 points, so by setting *thresh*$_2$ ≥ 16p in the paragraph macro, very short last lines, like might occur with the "etc." in the previous paragraph, can be prevented should the progressive penalty allow them.

.overrunpenalty 1 0 0.5c attempts to prevent lines shorter than 0.5 centimeter, but does not assess a progressive penalty. The value of *pen* is not important in this case, but it must be greater than zero.

.wsmark 0 | 1 *Default:* 0

Places a number or character in the right margin to indicate the word space bin of the line. A 1 or no argument switches it on, 0 switches it off. This is useful when fixing problem paragraphs.

.wsmin *lwr* *Default:* 0

Defines a soft minimum word space in percent of the desired space size. The request **.wsmin 71** sets the minimum space size to 71% of

the desired space size. The value of *lwr* must be greater than 100 × *minss* / *ss* for there to be any effect.

With non-justified paragraphs, **wsmin** strongly discourages lines shorter than *lwr* percent of full measure.

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

Similar to **.wsmark**, but only flags lines that need attention; optionally writes a message to `stderr`. If a line has word spaces smaller than *lwr* percent or greater than *upr* percent of the desired size and *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 retained.

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

.wswarn 2 75 138 marks all lines with spaces smaller than 75% or greater than 138% of nominal (the farther half of the "tight" and "loose" classes), and writes a message to `stderr`:

```
wswarn: wordspace bin 2 ratio 0.741372 on page 9
wswarn: wordspace bin 2 ratio 0.712140 on page 11
wswarn: wordspace bin 8 ratio 1.425903 on page 15
```

If both **.wswarn** and **.wsmark** are in effect at the same time, **.wsmark** takes precedence for marking the lines in the margin, but the messages sent to `stdout` originate from **.wswarn**. 2

.letcalc *n* *Default:* 0

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

n	Description
0	Heirloom mode
1	Letter adjustment last ("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
21	As 2, but applies letter spacing first, then glyph scaling
22	As 2, but applies glyph scaling first, then letter spacing
31	As 3, but applies letter spacing first, then glyph scaling
32	As 3, but applies glyph scaling first, then letter spacing

Method 0 executes the Heirloom code with no changes; does not support any of the new features.

Method 1 is based on the Heirloom approach, and adapted to work with the new features. Lines that have "good" word spaces as defined by the **.wrdspc** request do not have letter adjustment applied. Of the new methods, it has the least glyph distortion, but the least improvement in word space uniformity and line break quality.

Method 2 applies all letter adjustment before adjusting the word spaces. This method is among the best for word space uniformity and line break quality, but it has 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 letters and spaces. Its line breaking characteristics are similar to Method 2.

Methods 0, 1, 2, 3, and 4 apply dynamic letter spacing and glyph scaling simultaneously. Methods 1 through 4 apply them in proportion to the values set with **.letadj**. Method 0 applies them equally as long as equal amounts are available.

Letter adjustment is switched on or off with the **.letadj** request. As with the Heirloom code, **.letadj** ***lspmin lshmin letss lspmax lshmax*** turns letter adjustment on, 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.

Letter Adjustment Feature Support

letcalc	letpen	adjletpen	letthresh	letstren
0	-	-	-	-
1	Y	Y	-	Y
2	Y	Y	Y	Y
3	Y	Y	Y	Y
4	Y	Y	Y	Y
21	Y	Y	Y	Y
22	Y	Y	Y	Y
31	Y	Y	Y	Y
32	Y	Y	Y	Y

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

Defines how the letter adjustment penalty is to be applied.

If *pen* = 0, the effects of letter adjustment are not taken into account when determining the line breaks, but letter adjustment is applied when the paragraph is output. However, the amount of letter shrink is still used to determine whether the text will fit on the line, so the effect is not the same as switching letter adjustment off with **.letadj** .

If *pen* = 1, the effects of letter adjustment are taken into account when determining line breaks. Letter adjustment is used freely within the constraints established by **.letadj**. The paragraph will have a strong tendency to use the maximum amount of letter adjustment.

If *pen* ≥ 2, the effects of letter adjustment are taken into account when determining the line breaks, and a penalty is applied to discourage excessive use of letter adjustment. A penalty of reference value *pen* is applied based on the reference threshold values of *shrinkthresh* and *stretchthresh*. **.letpen 10 0.5 0.65** applies a penalty of 10 when the total amount of letter adjustment is 0.5% if the line is to be shrunk, or 0.65% if the line is to be stretched. The penalty is progressive: it is zero when no letter adjustment is applied, and it is greater than *pen* when the amount of letter adjustment is greater than the threshold values.

.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 is applied. The behavior of this request depends on the arguments to **.letpen**. If **.letpen**'s penalty is ≥ 2, and the letter adjustment of one line is shrunk by *pct* times **.letpen**'s *shrinkthresh*, and the other line is stretched by *pct* times **.letpen**'s *stretchthresh*, reference penalty *pen* is applied. For letter adjustment greater than the threshold, the applied penalty increases sharply. If letpen's penalty is less than 2, **.adjletpen** has no effect.

For **.letpen 10 0.5 0.65** and **.adjletpen 30 100**, a penalty of 30 is applied if the letter adjustment of one line is shrunk more than 0.5% and the other is stretched more than 0.65%. The **.adjletpen** penalty is in addition to the penalty applied by **.letpen**.

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

Changes the dynamic glyph scaling 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 width.

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

Changes the dynamic letter spacing limits to *lwr* and *upr*. Equivalent to the request **.letadj *lwr* x x *upr* x**, but is less error prone and does not throw a "bad number" message. 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 of letter adjustment, expressed in percent; the normal strength 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 maintain the line's inherent proportion of total character length to total space length until all letter adjustment has been used up.

.letthresh *lwr upr* *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. *lwr* establishes the smallest space size in this region, *upr* the largest. *lwr* must be 100 or less, *upr* must be 100 or greater.

.letthresh 83 125 applies no letter adjustment if the word spaces for the line are between 83% and 125% of the desired size.

.letthresh 100 100 applies letter adjustment to all lines.

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
.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
.letlwr	f	%	letter adjustment penalty lower reference
.letpen	i		letter adjustment penalty
.letstren	f	%	letter adjustment strength multiplier
.letthreshlwr	f	%	exclusion zone lower threshold
.letthreshupr	f	%	exclusion zone upper threshold
.letupr	f	%	letter adjustment penalty upper reference

The following examples demonstrate a variety of paragraph composition styles that can be obtained by modifying the paragraph control and letter adjust parameters. Justified paragraphs (**.ad b**) are assumed except as indicated. `CRR` refers to Crimson Roman, as used for this text.

To simulate the appearance of the Monotype's standard line breaking, from the early 1900s through the end of its production:

```
.fspacewidth CRR 308
.ss 12
.minss 8
.padj 0
.letadj
```

Tight spacing can also be simulated:

```
.fspacewidth CRR 308
.ss 8 0
.minss
.padj 0
.letadj
```

Note that **.wsmark** and **.wswarn** will report different values when *ss* = 8, even though the font's space size remains the same. Referring to the adjustment ratios of Table 1, the acceptable bins are numbered 5 through C; the desired space size is now 8 / 12 ≈ 0.667, so the upper ratio is 1.5 / 0.667 ≈ 2.25, or a little larger than bin C.

The first patents for the Monotype, filed by Tolbert Lanston in 1885, described a method of justifying lines by inserting space between all the type elements on the line. A second method distributed letter spacing in proportion to the character widths, and a third inserted additional space around punctuation. The production models, however, did not

justify lines this way; they stretched the spaces to fill the line. Letter spacing remained part of the design, but was done by manual control. The equal letter spacing justification method might have looked something like:

```
.fspacewidth CRR 308
.padj 0
.ss 12
.minss
.letcalc 4
.letadj 100 100 12 120 100
```

and for tight spacing:

```
.fspacewidth CRR 308
.padj 0
.ss 8 0
.minss
.letcalc 4
.letadj 100 100 12 120 100
```

troff appeared in the 1970s. It used a single-line composition method like the one taught an apprentice hand compositor:

```
.fspacewidth CRR 333
.ss 12
.minss
.padj 0
.letadj
```

troff could also do tight spacing. The basis for the space size was fixed at one-third of an em, but the effective value could be made smaller:

```
.fspacewidth CRR 333
.ss 7
.minss
.padj 0
.letadj
```

TEX introduced paragraph-at-once adjustment ca. 1980, along with features to control hyphenation and improve paragraph composition, but there was no letter adjustment support:

```
.fspacewidth CRR 333
.ss 12
.minss 8
.padj
.letadj
.wscalc 5
.wrdspc 66.7 150\"  wscalc 5 sets these automatically
.adjpenalty 100 50\" so there is no need to list them
.hypp 25 100 70\"   unless changing the default value
.lastlinestretch\"                " "
.linepenalty 1\"                  " "
```

groff was introduced in the early 1990s. An open source version of *troff*, it retained the "greedy" single-line composition method, but offered an improved way of selecting break points, it could shrink word spaces as well as stretch them, and it supported the font's native space width. The *groff* model is thus similar to the Monotype single-line method.

pdfTEX added letter adjustment to TEX ca. 2000. The original implementation supported glyph scaling and dynamic letter spacing, and applied all letter adjustment before adjusting the space size. TEX also began to support the font's native spacewidth:

```
.spacewidth
.ss 12
.minss 8
.padj
.wscalc 5
.letadj 0 98 12 0 102\" the recommendation was ±2%
.letcalc 2
.letpen 1
.letthresh 100 100
```

The Heirloom enhancements were made to *troff* around 2005. These changes added many typographic features, including paragraph-at-once justification and dynamic letter adjustment. Letter adjustment is applied only if the word spaces are larger than a specified threshold or if the line would otherwise be too long to fit.

```
.spacewidth
.ss 12
.minss 8
.padj
.wscalc 0
.letcalc 0
.letadj 99 99 16 101 101
.hypp 40 40 40\" default is 0 0 0
```

The new features described here take some of the previous concepts a bit further. For example, if we prefer the word spaces to be between 75% and 140% of the font's nominal size; do not want to apply any letter adjustment to word spaces that are in the "decent" fitness class; want to apply letter adjustment before adjusting the spaces; take the effects of letter adjustment into account when determining line breaks, while discouraging excessive use of letter adjustment; and discourage adjacent lines with visually incompatible shrink and stretch:

```
.spacewidth
.ss 12
.minss 8
.padj
.wscalc 4
.wrdspc 75 140
.letadj 99 99 12 101 101
.letthresh 83 125
.letcalc 2
.letpen 5 0.6 0.7
.adjletpen 5 100
.hypp 50 50 50
```

If we want to use a least-squares word space penalty, prefer spaces between 75% and 133% of nominal size, apply letter adjustment only to lines with spaces outside the preferred range, choose line breaks with the effects of letter adjustment factored in but not penalized, moderately discourage adjacent lines that straddle the "decent" fitness class, discourage last lines that are shorter than 25% of full measure and prevent lines shorter than 18 points, make the paragraph one line shorter, and flag lines that are in or worse than the farther half of the "tight" or "loose" fitness classes and write a message to the terminal:

```
.wscalc 2
.wrdspc 75 133
.letcalc 1
.letpen 1
.adjpenalty 50 83 125
.overrunpenalty 50 25 18p
.looseness -1
.wswarn 2 75 137.5
```

With non-justified paragraphs in paragraph mode, some of the paragraph controls are supported, but letter adjustment is not. If we want a ragged right paragraph, prefer the lines to be at least 85% of full measure, want all lines to be at least 80% of full measure, use a cubic penalty curve, and discourage overruns:

```
.padj
.ad l
.wrdspc 85 150
.wsmin 80
.wscalc 3
.overrunpenalty 50 25 0.25i
```

Table 1: Bin classes and space adjustment ratios

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

Notes:

1. TEX fitness class 1, "Decent," spans bins 4, 5, and 6. These class numbers and descriptions are from the Knuth-Plass paper. The class numbers are reversed in TEX82, so that Very loose = 0, Loose = 1, Decent = 2, and Tight = 3. In the *troff* typographical additions, Decent = 0, looser classes have positive numbers, and tighter classes have negative numbers. The description "Awful" is from the TEX Book; "Awful" is orders of magnitude better than the "Awful bad" of TEX82. The Very Tight class was possible on Monotypes beginning in 1925, but is not supported by TEX.
2. The Normalized Adjustment Ratio *r* shown here corresponds to the TEX range of word spaces: 66.7% to 150% of the desired size (**.wrdspc 66.7 150**). Changing the **.wrdspc** range changes the value of *r* accordingly. For example, **.wrdspc 75 137.5** scales *r* so that $r = -1.0$ when the raw adjustment ratio $R = 0.75$, and $r = +1.0$ when $R = 1.375$; this causes the algorithm to try to reduce word space variation. The raw adjustment ratio and bin classes are not affected by **.wrdspc**. See also Figures 1 and 4.

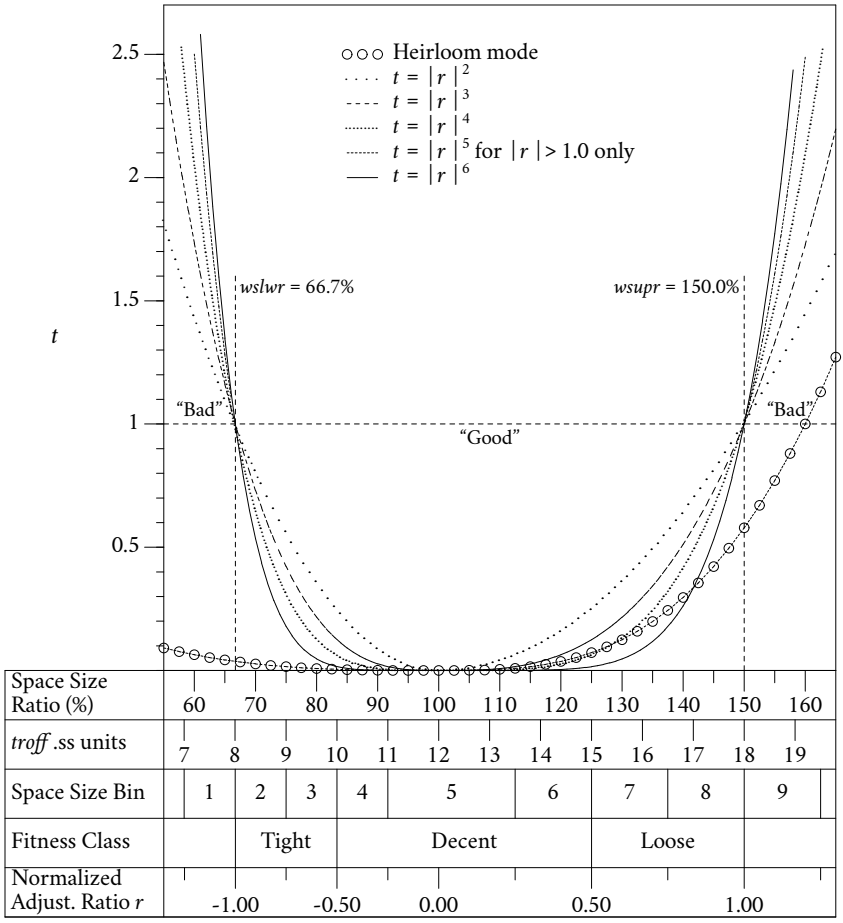

Figure 1: Default relationships for the space size adjustment ratio, *troff* space size units, space size bins, fitness classes, normalized adjustment ratio *r*, and penalty *t*. The defaults are: **.ss 12**, **.wrdspc 66.7 150**, **.adjpenalty *n* 50** (alternately **.adjpenalty *n* 83.3 125**). The word space penalty *t* is calculated from the normalized adjustment ratio, which is established by the arguments to the **.wrdspc** request. Many of the other requests are based on the percent of nominal space size. The various penalties input by the user are divided by 100 and added to *t* as appropriate, shifting the curve vertically.

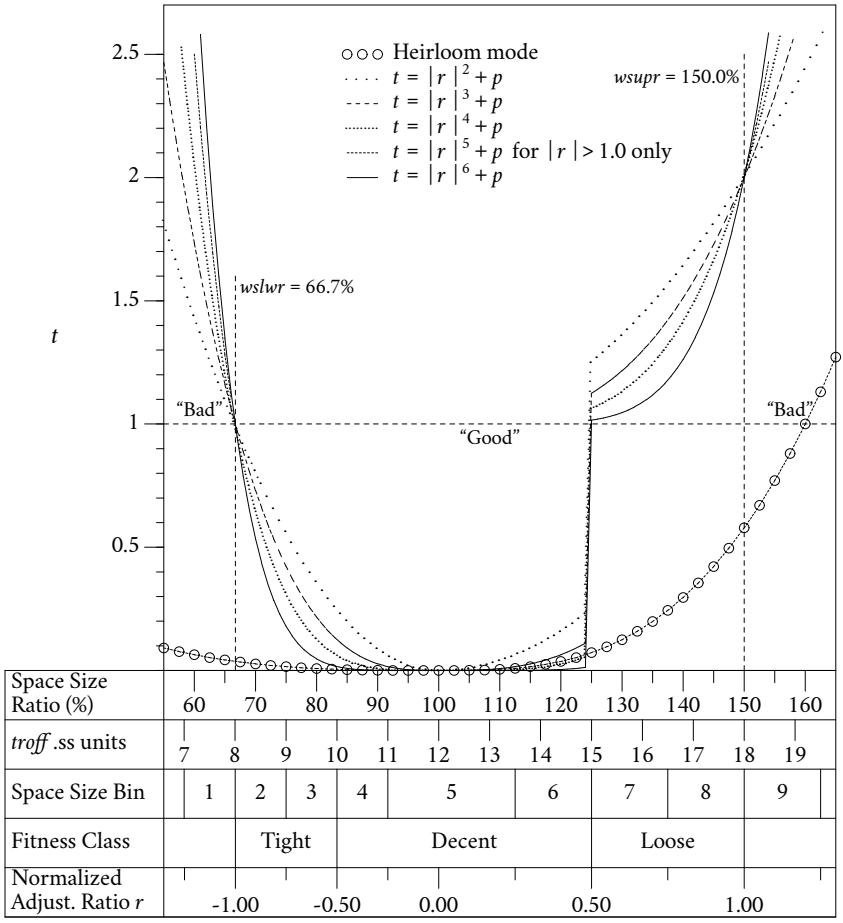

Figure 2: Effect of **.adjpenalty 100 50** on the penalty curve of Figure 1 for a line following a "tight" line. If the second line is "loose," it is assessed an additional penalty of 1.0 (the user's input value of 100 divided by 100), but if it is "tight" or "decent," it is assessed the normal penalty. Heirloom mode does not support **.adjpenalty**.

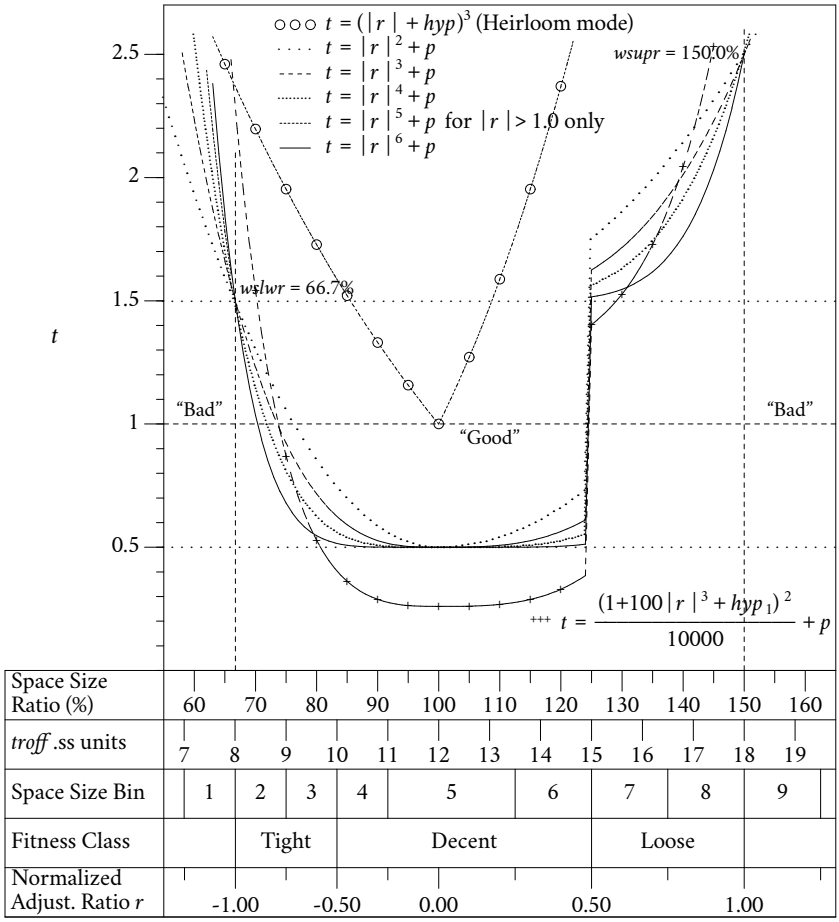

Figure 3: Penalty curves for a hyphenated line following a "tight" line, with user inputs **.adjpenalty 100 50** and **hypp 50**. The hyphenation penalty is applied to all hyphenated lines. In the new modes, all penalties are added together, shifting the penalty curve vertically. In the Heirloom and KP-81 penalty curves, the hyphenation penalty is included in the word space penalty calculation, causing the shape of the penalty curve to change. The Heirloom hyphenation penalty is also twice as strong for any given input. The labels for "good" and "bad" are in their original positions: horizontally as they relate to *wslwr* and *wsupr*, and vertically as they relate to individual characteristics, not the sum of the penalties.

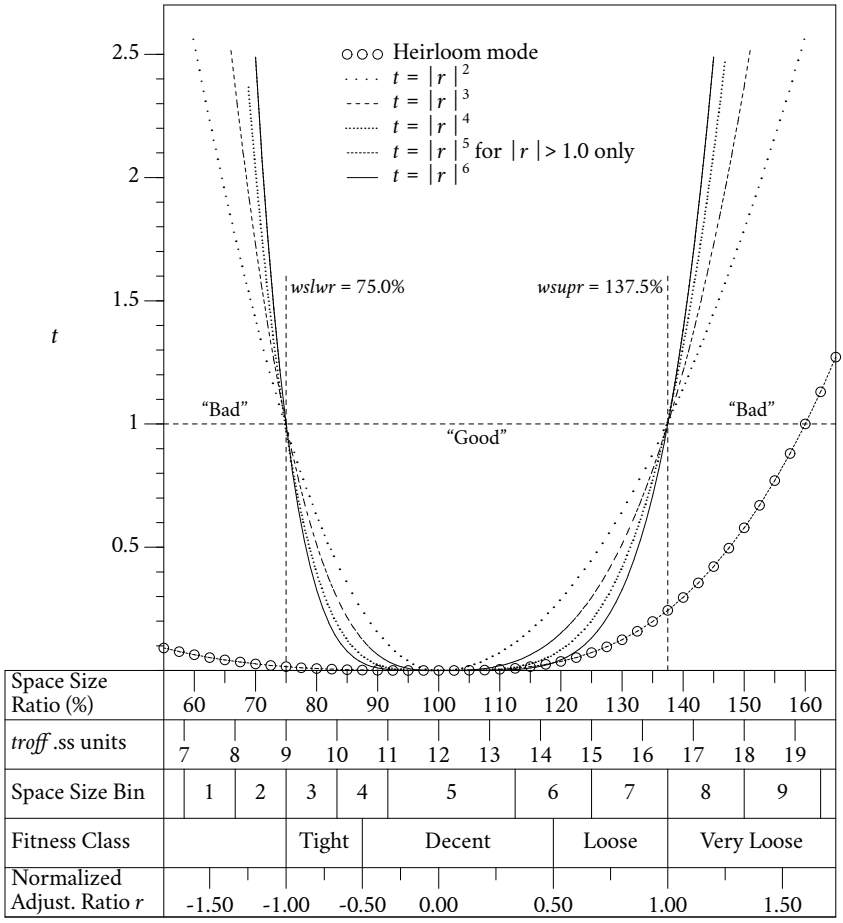

Figure 4: Effect of setting **.wrdspc 75 137.5** on the word space penalty curve *t*, the normalized adjustment ratio *r*, and fitness classes. In this example the fitness classes have been specified with **.adjpenalty *n* 50**, so they have been scaled relative to the normalized space adjustment ratio. If they had been specified with **.adjpenalty *n* 83.3 125**, they would have remained fixed in the same locations as in Figures 1, 2, and 3. Heirloom mode (**.wscalc 0**) does not support the **.wrdspc** request, but the Heirloom penalty calculation method (**.wscalc 1**) does.

The main text is set in a modified version of Crimson, 11.75/15 x 26, with dynamic letter spacing from 98.5% to 101.5%; glyph scaling was not used. The constant width typeface is Inconsolata.