TypoG Examples

The section numbers correspond to the subsections of section 3 in the official documentation of package typog.

Contents

Examples

Unless otherwise noted the font used in the examples is ›Inter‹.

1 Setup and Reconfiguration

`\typogget`

`breakpenalty` ↦ 50
`ligaturekern` ↦ $^{31}/_{1000}$ em
`lowercaselabelitemadjustments` ↦ -0.6668pt, 0.0pt, 0.9998pt, 0.0pt
`lowerslash` ↦ 0.0pt
`mathitalicscorrection` ↦ 0.4mu
`raisecapitaldash` ↦ 0.0pt
`raisecapitalguillemets` ↦ 0.0pt
`raisecapitalhyphen` ↦ 0.0pt
`raisecapitaltimes` ↦ 0.0pt
`raisefiguredash` ↦ 0.0pt
`raiseguillemets` ↦ 0.0pt
`raiseinvertedmarks` ↦ 0.0pt, 0.0pt, 0.0pt
`shrinklimits` ↦ 5, 10, 20
`slashkern` ↦ 0.4999pt
`stretchlimits` ↦ 5, 10, 20
`textitalicscorrection` ↦ 0.19998pt
`trackingttspacing` ↦ 300, 90, 60
`uppercaselabelitemadjustments` ↦ 0.0pt, 0.9998pt, 1.49957pt, 0.9998pt

`\typoggetnth`

`lowercaselabelitemadjustments`: (1) -0.6668pt, (2) 0.0pt, (3) 0.9998pt, and (4) 0.0pt.
Same elements now accessed from the right-hand-side, i. e. with negative indices: (–4) -0.6668pt, (–3) 0.0pt, (–2) 0.9998pt, and (–1) 0.0pt.

`raiseinvertedmarks`: (1) 0.0pt, (2) 0.0pt, and (3) 0.0pt.

`shrinklimits`: (1) 5, (2) 10, and (3) 20.

`stretchlimits`: (1) 5, (2) 10, and (3) 20.

`trackingttspacing`: (1) 300, (2) 90, and (3) 60.

`uppercaselabelitemadjustments`: (1) 0.0pt, (2) 0.9998pt, (3) 1.49957pt, and (4) 0.9998pt.

2 Information

`\fontsizeinfo` – At this point of the document, the font size and the line spacing are 10.6/13 (w/o units). For footnotes however, the current sizes are[1] 8.5 pt/ 9.5 pt.

Next we show a comparison of different font sizes and line spacings decorated with the results of `\fontsizeinfo`.

Different font sizes and line spacings exemplified with the Merriweather font.

Macro `\baselineskip` is a length command which specifies the minimum space between the bottom of two successive lines in a paragraph. Its value may be automatically reset by LaTeX, for example, by font changes in the text.

Merriweather 8.5/12

Macro `\baselineskip` is a length command which specifies the minimum space between the bottom of two successive lines in a paragraph. Its value may be automatically reset by LaTeX, for example, by font changes in the text.

Merriweather 10/12

Macro `\baselineskip` is a length command which specifies the minimum space between the bottom of two successive lines in a paragraph. Its value may be automatically reset by LaTeX, for example, by font changes in the text.

Merriweather 10/13.5

Starred form eats spaces? 10/12.

3 Hyphenation

Line-break behavior

`\mbox+\breakpoint*`
(pre-)
Hilbert
space

`\breakpoint*`
(pre-
)
Hilbert
space

`\breakpoint`
(pre-
)
Hil-
bert
space

Starred form eats spaces? ab. Unstarred: ab.

[1] This is the footnote where we get the sizes from.

Set minimum hyphenation values for both `\lefthyphenmin` and `\righthyphenmin`: 6 and 6.

Set minimum hyphenation values for `\lefthyphenmin` and `\righthyphenmin` separately: 4 and 5.

Returned to the default values for `\lefthyphenmin` and `\righthyphenmin`: 2 and 3.

4 Disable/Break Ligatures

Macro	Result
n/a	fine affirmation of baffling flavors
`\nolig*`	fine affirmation of baffling flavors
`\nolig`	fine affirmation of baffling flavors
`\nolig*[75]`	office

Line-break behavior

`\nolig*` bijection

`\nolig` bijection

Starred form eats spaces? fi, fi.

5 Manual Italic Correction

Text Mode. The italic correction of the current font is 0.0pt/pt.

We demonstrate the effect of `\itcorr` with a pair of bookends: uncorrected italics: 𝐼𝑋, TEX-corrected (`\/`): 𝐼𝑋, and `\itcorr{7}`: 𝐼𝑋.

Correction 0: 𝐼; corr. 3: 𝐼, 𝐼 (starred); corr. –6: 𝐼.

Mathematical Mode. Uncorrected: [f], corrected: [f]
Correction 0: $\perp$; corr. 3: $\perp\!\!\!\perp$; corr. –6: $\perp\!\!\!\perp$.

6 Apply Extra Kerning

6.1 Slash

The slash with some extra space (`slashkern`=0.4999pt) around it can be helpful for certain pairs, as for example years or names.

Macro	Local Settings	Result
n/a	n/a	1991/1992, New York/NY, Korringa/Kohn/Rostoker
`\kernedslash`	n/a	1991/1992, New York/NY, Korringa/Kohn/Rostoker
`\kernedslash`	`lowerslash=.1em, slashkern=-.0333em`	BARDEEN/COOPER/SHRIEFFER
`\kernedslash`	`lowerslash=.1em, slashkern=0pt`	abc/def, uvw/jkl

Line-break behavior

`\kernedslash*`
1991/1992,
New York/NY,
Korringa/Kohn/Rostoker

`\kernedslash`
1991/
1992,
New York/
NY,
Korringa/
Kohn/
Rostoker

`\kernedslash*\nobreak`
1991/1992,
New York/NY,
Korringa/Kohn/Rostoker

`\allowhyphenation\kernedslash`
1991/
1992,
New York/
NY,
Ko-
r-
ringa/
Kohn/
Ros-
toker

Starred form eats spaces? p / q.

6.2 Hyphen

Uncorrected

K-vector space, *g*-factor, *f*-function

Corrected

K-vector space, *g*-factor, *f*-function

Line-break behavior

hyphen '-'	`\hyp`	`\kernedhyphen*`	`\kernedhyphen`
self-energy	self-	self-energy	self-
			en-
	en-		ergy
	ergy		

If a `\kernedhyphen` goes astray in a math environment, it decays to an ordinary minus with appropriate kerning: $G–V$.

7 Raise Selected Characters

7.1 Capital Hyphen

With the standard hyphen we get

NMR-Spektroskopie, SI-Einheit, *G*-Modul, and *K*-Vektorraum,

whereas with raising the hyphen by 0.6667pt when calling `\capitalhyphen`, we arrive at

NMR-Spektroskopie, SI-Einheit, *G*-Modul, and *K*-Vektorraum (even better with `\kernedhyphen` and the star-option for the correct raise-amount: *K*-Vektorraum).

Line-break behavior

`\capitalhyphen*`	`\capitalhyphen`
NMR-Spektroskopie	NMR-
	Spek-
	tro-
	sko-
	pie

Starred form eats spaces? V-W.

7.2 Capital Dash

Compare the result of plain `\textendash`

A–M, N–Z, C1 – C4, LEED – STM

with `\capitaldash`:

A–M, N–Z, C1 – C4, LEED – STM

where the en-dash has been raised by $^{75}/_{1000}$ em.
Starred form eats spaces? V–W.

7.3 Number Dash

Compare the result of plain `\textendash`

3–5, 81–82, 485–491

with `\figuredash`:

3–5, 81–82, 485–491

where the en-dash has been raised by 0.6667pt.
Line-break behavior

`\figuredash*`	`\figuredash`
3–5,	3–
81–82,	5,
485–491	81–
	82,
	485–
	491

Starred form eats spaces? 44–55.

7.4 Multiplication Sign – Times "×"

The problem with a too-low multiplication sign arises for example with matrices of a given, specific size.
Uncorrected

LR-mode: 2×2-matrix, $N \times M$-matrix
Math-mode: 2×2-matrix, $N \times M$-matrix

and corrected

LR-mode: 2×2-matrix, $N \times M$-matrix
Math-mode: 2×2-matrix, $N \times M$-matrix.

7.5 Guillemets

We again compare the default implementation with the adjusted one.

›Use single quotes for a first quotation.‹
»Use double quotes for quotations within quotations.«
›1‹, ›2‹, ›3‹.
»ABC«, »MN«, »XYZ«.

Corrected by raising the glyphs by $^{50}/_{1000}$ em and $^{100}/_{1000}$ em, respectively:

›Use single quotes for a first quotation.‹
»Use double quotes for quotations within quotations.«
›1‹, ›2‹, ›3‹.
»ABC«, »MN«, »XYZ«.

And the same using French typographic conventions:

‹ Use single quotes for a first quotation. ›
« Use double quotes for quotations within quotations. »
‹ 1 ›, ‹ 2 ›, ‹ 3 ›.
« ABC », « MN », « XYZ ».

Line-break behavior

›re-	›re-	»re-	»re-
la-	la-	la-	la-
tion‹	tion‹	tion«	tion«
‹ re-	‹ re-	« re-	« re-
la-	la-	la-	la-
tion ›	tion ›	tion »	tion »

7.6 Inverted Exclamation Mark and Inverted Question Mark

Without TypoG support: g¡jk¿y ¡E! ¿Q?

All-caps versus small-caps:

CARMEN
¿CÓMO ESTÁS, JOSÉ?
JOSÉ
¡BIEN! ¿Y TÚ?
CARMEN
¡BIEN TAMBIÉN!

CARMEN
¿cómo estás, josé?
JOSÉ
¡bien! ¿y tú?
CARMEN
¡bien también!

Local settings are {5.0pt, 0.0pt, 0.00002pt}.

Primaries: ?¿? and !¡!

Secondaries – indexed with 2: ¿2Más? ¡2Por venir!

Tertiaries – indexed with 3: ¿3Continuemos? ¡3Con el texto!

8 Label Items

The current configurations for the uppercase and lowercase adjustments are {0.0pt, 0.9998pt, 1.49957pt, 0.9998pt} and {-0.6668pt, 0.0pt, 0.9998pt, 0.0pt}, respectively.

A•B–E∗G·H
a•b–c∗e·g

For the following nested lists we adjust levels one and three for uppercase and levels two and four for lowercase.

- A
- B
- C
  - d
  - e
  - f
    - G
    - H
    - I
      - j
      - k
      - l

The left example shows the result of `\typogadjuststairs{.25pt}{11}{ABC}` and the one on the right hand side just uses the same parameters except for the sample 'ace', this is, only lowercase letters.

For this document the calls `\typoguppercaseadjustcheck{ABCDEFGHIJKL-MNOPQRSTUVWXYZ}` and `\typoglowercaseadjustcheck{acegmnopqrsuvwxyz}` yield '•–∗·' and '•–∗·', where the adjustments in effect are `{0.0pt, 0.9998pt, 1.49957pt, 0.9998pt}` and `{-0.6668pt, 0.0pt, 0.9998pt, 0.0pt}`, respectively.

9 Align Last Line

9.1 Last Line Ragged Left/Flush Right

Example 9.1 is a typical use of environment `lastlineraggedleftpar`: A narrow paragraph gets typeset with full justification and put `\flushright` against the right margin as a whole.

The layout may look more coherent if the last lines is moved to the right margin, too.

Example 9.1 – Typeset a justified paragraph flushright and let macro `\lastlineraggedleft` shift the last line over to the right-hand side.

We still haven't discussed the special trick that allows the final line of a paragraph to be shorter than the others. Just before TEX begins to choose breakpoints, it does two important things: [...]

The sample text was taken from The TEXbook [1, p. 99n].

The font used in this example is Inter regular, 9/11.

9.2 Last Line Centered

The situation shown in Ex. 9.2 is more widespread than Ex. 9.1 because centered tables and figures are quite common. Their caption parboxes are centered too, which is where a centered last line might fortify the layout.

Another possible use of environment `lastlinecenteredpar` are the final lines of chapters – in particular if the chapters' ends are marked with centered dingbats.

Example 9.2 – Typeset a justified paragraph that is centered. This very caption uses `lastlinecenteredpar` to have its last line centered as well. Moreover, we put a nifty asterisk centered at the bottom of the sample text.

We still haven't discussed the special trick that allows the final line of a paragraph to be shorter than the others. Just before TEX begins to choose breakpoints, it does two important things: [...]

*

The sample text was taken from The TEXbook [1, p. 99n].

The font used in this example is Inter regular, 9/11.

---

10 Fill Last Line

Example 10.1 – Top example: Typeset a paragraph without correction of the last line. Middle example: Paragraph corrected with `covernextindentpar`. We set a `\parindent` of 25pt in both parboxes and we *must* increase the amount of glue in the paragraph to reduce the penalty of stretching the last line under a `\fussy` setting. For the samples below, we have chosen `\slightlysloppy[2]`. The ›Alternative‹, the bottom example, shows the effect of `tightspacing`; no extra sloppyness is required there.

We still haven't discussed the special trick that allows the final line of a paragraph to be shorter than the others. Just before TEX begins to choose breakpoints, it does two important things: [...] abcd efgh ijkl mnop qrst uvwx yz12 3456

We still haven't discussed the special trick that allows the final line of a paragraph to be shorter than the others. Just before TEX begins to choose breakpoints, it does two important things: [...] abcd efgh ijkl mnop qrst uvwx yz12 3456

Alternative...

We still haven't discussed the special trick that allows the final line of a paragraph to be shorter than the others. Just before TEX begins to choose breakpoints, it does two important things: [...] abcd efgh ijkl mnop qrst uvwx yz12 3456

The sample text was taken from The TEXbook [1, p. 99n].

The font used in this example is Inter regular, 9/11.

In this paragraph the value of `\lastlinefit` was increased to 1000. Thus, the spacing of its last line is equal to that of the next-to-last line. The effect can be double-checked with log-file, *typog-example.log*, as this text was wrapped in a `typoginspect` environment with id `lastlinefitpar`.

The value of `\lastlinefit` is reset to 0 in the following paragraph.

Example 10.2 – Same comparison as the previous example, but for a small linewidth and zippo `\parindent`. The left-hand side sample is uncorrected, the right-hand side features `\covernextindentpar`. The sloppyness level is 2 for both samples.

We still haven't discussed the special trick that allows the final line of a paragraph to be shorter than the others. Just before TEX begins to choose breakpoints, it does two important things: [...] abcd efgh ijkl mnop qrst uvwx yz12 3456

We still haven't discussed the special trick that allows the final line of a paragraph to be shorter than the others. Just before TEX begins to choose breakpoints, it does two important things: [...] abcd efgh ijkl mnop qrst uvwx yz12 3456

The sample text was taken from The TEXbook [1, p. 99n].

The font used in this example is Inter regular, 9/11.

Example 10.3 – Sample 1: Typeset a paragraph without correction of the last line. Sample 2: Paragraph corrected with `\openlastlinepar`. – Disappointing! Sample 3: Same using macro `\prolongpar`. Sample 4: Alternative solution that simply increases the tracking by $^{2}/_{1000}$ em with `setfonttracking`. Sample 5: Alternative solution that increases the spacing with `loosespacing`.

We still haven't discussed the special trick that allows the final line of a paragraph to be shorter than the others. Just before TEX begins to choose breakpoints, it does two important things: [...] abcd efgh ijkl mnop qrst uvwx yz12 3456

We still haven't discussed the special trick that allows the final line of a paragraph to be shorter than the others. Just before TEX begins to choose breakpoints, it does two important things: [...] abcd efgh ijkl mnop qrst uvwx yz12 3456

We still haven't discussed the special trick that allows the final line of a paragraph to be shorter than the others. Just before TEX begins to choose breakpoints, it does two important things: [...] abcd efgh ijkl mnop qrst uvwx yz12 3456

Alternatives...

We still haven't discussed the special trick that allows the final line of a paragraph to be shorter than the others. Just before TEX begins to choose breakpoints, it does two important things: [...] abcd efgh ijkl mnop qrst uvwx yz12 3456

We still haven't discussed the special trick that allows the final line of a paragraph to be shorter than the others. Just before TEX begins to choose breakpoints, it does two important things: [...] abcd efgh ijkl mnop qrst uvwx yz12 3456

The sample text was taken from The TEXbook [1, p. 99n].

The font used in this example is Inter regular, 9/11.

11 Spacing

11.1 Narrow/Wide Space

The current font's parameters are shown in table 1.[2]

Table 1: Important `\fontdimen` values of the current text font. The middle column (#) states the number of the fontdimen.

Name	#	Value
Interword space	2	2.97917pt
Interword stretch	3	1.32999pt
Interword shrink	4	1.38324pt
Extra space	7	0.37453pt

Compare	some text with spaces	default space, natural glue
with	some text with spaces	`\narrowspace`, natural glue
	some text with spaces	`\narrowspace`, tight box
	some text with spaces	`\narrowspace`, spread 5pt
and again	some text with spaces	default space, natural glue
with	some text with spaces	`\widespace`, natural glue
	some text with spaces	`\widespace`, tight box
	some text with spaces	`\widespace`, spread 5pt

Starred form eats spaces? Narrow Space. Wide Space.

11.2 Looser/Tighter

Examples 11.1 and 11.2 show `tightspacing` and `loosespacing` at work.

[2] For a concise and understandable explanation of the plethora of font parameters consult David Carlisle's excellent post on StackExchange: What Do Different Fontdimennum Mean.

Example 11.1 – Both parboxes are typeset with `\slightlysloppy[3]`, the left one with default spacing, the right one with `tightspacing[1]`.

If you want to avoid overfull boxes at all costs without trying to fix them manually, you might be tempted to set `tolerance=10000`; this allows arbitrarily bad lines to be acceptable in tough situations. But infinite tolerance is a bad idea, because TEX doesn't distinguish between terribly bad and preposterously horrible lines. Indeed, a tolerance of 10000 encourages TEX to concentrate all the badness in one place, making one truly unsightly line instead of two moderately bad ones, because a single "write-off" produces fewest total demerits according to the rules.

If you want to avoid overfull boxes at all costs without trying to fix them manually, you might be tempted to set `tolerance=10000`; this allows arbitrarily bad lines to be acceptable in tough situations. But infinite tolerance is a bad idea, because TEX doesn't distinguish between terribly bad and preposterously horrible lines. Indeed, a tolerance of 10000 encourages TEX to concentrate all the badness in one place, making one truly unsightly line instead of two moderately bad ones, because a single "write-off" produces fewest total demerits according to the rules.

The sample text was taken from The TEXbook [1, p. 107].

The font used in this example is Inter regular, 9/11.

Example 11.2 – Both parboxes are typeset with `\slightlysloppy[3]`, the left one with default spacing, the right one with `loosespacing[2]`.

If you want to avoid overfull boxes at all costs without trying to fix them manually, you might be tempted to set `tolerance=10000`; this allows arbitrarily bad lines to be acceptable in tough situations. But infinite tolerance is a bad idea, because TEX doesn't distinguish between terribly bad and preposterously horrible lines. Indeed, a tolerance of 10000 encourages TEX to concentrate all the badness in one place, making one truly unsightly line instead of two moderately bad ones, because a single "write-off" produces fewest total demerits according to the rules.

If you want to avoid overfull boxes at all costs without trying to fix them manually, you might be tempted to set `tolerance=10000`; this allows arbitrarily bad lines to be acceptable in tough situations. But infinite tolerance is a bad idea, because TEX doesn't distinguish between terribly bad and preposterously horrible lines. Indeed, a tolerance of 10000 encourages TEX to concentrate all the badness in one place, making one truly unsightly line instead of two moderately bad ones, because a single "write-off" produces fewest total demerits according to the rules.

The sample text was taken from The TEXbook [1, p. 107].

The font used in this example is Inter regular, 9/11.

12 Microtype Front-End

12.1 Tracking

Example 12.1 – Use microtype to change the font tracking. The sample on the left-hand side shows neutral tracking. The one on the right-hand side received an extra tracking of $^{7}/_{1000}$ em.

This sentence contains an explicit call to `\textls` with an optional argument of (+200) to D E M O N S T R A T E that this macro still works inside of `setfonttracking`. Apart from that it is just some more text to exercise the macro. Well, the explicit letterspacing example is particularly ugly.

This sentence contains an explicit call to `\textls` with an optional argument of (+200) to D E M O N S T R A T E that this macro still works inside of `setfonttracking`. Apart from that it is just some more text to exercise the macro. Well, the explicit letterspacing example is particularly ugly.

The font used in this example is Inter regular, 9/11.

Example 12.2 – Check how font changes (serif, serif italics, small-caps, sans serif, typewriter) interfere with the interword spacing. The left sample has no tracking changes applied and serves as a reference, whereas the right sample got an extra tracking of $^{1}/_{1000}$ em. ¶ The switch from and to typewriter, i. e., constant-width fonts commonly is a source of spacing problems.

RM SF RM TT RM; RM *IT* RM; RM SC RM. Rm Sf Rm Tt Rm; Rm *It* Rm; Rm Sc Rm. rm sf rm tt rm; rm *it* rm; rm sc rm.

RM SF RM TT RM; RM *IT* RM; RM SC RM. Rm Sf Rm Tt Rm; Rm *It* Rm; Rm Sc Rm. rm sf rm tt rm; rm *it* rm; rm sc rm.

No contents: ⊥.

12.2 Font Expansion

No contents – `setfontshrink`: ⊥.
No contents – `setfontstretch`: ⊥.
No contents – `setfontexpand`: ⊥.
No contents – `nofontexpansion`: ⊥.

12.3 Character Protrusion

No contents – `nocharprotrusion`: ⊥.

Example 12.3 – Use microtype to stretch or shrink a font. The top sample uses `\setfontshrink` at level 3, the middle sample is the unchanged reference (which is allowed to shrink and expand), and the bottom sample utilizes `\setfontstretch` at level 2.

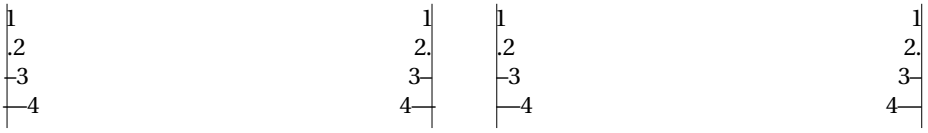
By default, all characters of a font are allowed to be stretched or shrunk by the same amount. However, it is also possible to limit the expansion of certain characters if they are more sensitive to deformation. This is the purpose of the `\SetExpansion` command.

By default, all characters of a font are allowed to be stretched or shrunk by the same amount. However, it is also possible to limit the expansion of certain characters if they are more sensitive to deformation. This is the purpose of the `\SetExpansion` command.

By default, all characters of a font are allowed to be stretched or shrunk by the same amount. However, it is also possible to limit the expansion of certain characters if they are more sensitive to deformation. This is the purpose of the `\SetExpansion` command.

The font used in this example is Erewhon regular, 9.6/11.

Example 12.4 – Comparison of the microtype feature "protrusion" (left-hand side) and `nocharprotrusion` (right-hand side).



13 Sloppy Paragraphs

Examples 13.1 and 13.2 put different amounts of “sloppiness” face to face.

Example 13.1 – Paragraphs typeset slightly sloppy: `\slightlysloppy` vs. `\fussy`. The left parbox is typeset with `\slightlysloppy` and *sloppiness* = 1, whereas the right sample features the well known `\fussy` setting. Both parboxes have a width of 180.0pt.

If you want to avoid overfull boxes at all costs without trying to fix them manually, you might be tempted to set `tolerance=10000`; this allows arbitrarily bad lines to be acceptable in tough situations. But infinite tolerance is a bad idea, because TEX doesn't distinguish between terribly bad and preposterously horrible lines. Indeed, a tolerance of 10000 encourages TEX to concentrate all the badness in one place, making one truly unsightly line instead of two moderately bad ones, because a single “write-off” produces fewest total demerits according to the rules.

If you want to avoid overfull boxes at all costs without trying to fix them manually, you might be tempted to set `tolerance=10000`; this allows arbitrarily bad lines to be acceptable in tough situations. But infinite tolerance is a bad idea, because TEX doesn't distinguish between terribly bad and preposterously horrible lines. Indeed, a tolerance of 10000 encourages TEX to concentrate all the badness in one place, making one truly unsightly line instead of two moderately bad ones, because a single “write-off” produces fewest total demerits according to the rules.

The sample text was taken from The TEXbook [1, p. 107].

The font used in this example is Inter regular, 9/11.

In conclusion all renderings of the text in Ex. 13.1 and Ex. 13.2 have their merits and their own flaws.

Example 13.2 – Paragraphs typeset slightly sloppy: `\slightlysloppy` vs. `\sloppy`. The left sample is features `\slightlysloppy` with *sloppiness* = 2, the right sample is typeset with `\sloppy`. Both parboxes have a width of 150.0pt.

If you want to avoid overfull boxes at all costs without trying to fix them manually, you might be tempted to set `tolerance=10000`; this allows arbitrarily bad lines to be acceptable in tough situations. But infinite tolerance is a bad idea, because TEX doesn't distinguish between terribly bad and preposterously horrible lines. Indeed, a tolerance of 10000 encourages TEX to concentrate all the badness in one place, making one truly unsightly line instead of two moderately bad ones, because a single "write-off" produces fewest total demerits according to the rules.

If you want to avoid overfull boxes at all costs without trying to fix them manually, you might be tempted to set `tolerance=10000`; this allows arbitrarily bad lines to be acceptable in tough situations. But infinite tolerance is a bad idea, because TEX doesn't distinguish between terribly bad and preposterously horrible lines. Indeed, a tolerance of 10000 encourages TEX to concentrate all the badness in one place, making one truly unsightly line instead of two moderately bad ones, because a single "write-off" produces fewest total demerits according to the rules.

The sample text was taken from The TEXbook [1, p. 107].

The font used in this example is Inter regular, 9/11.

14 Vertically Partially-Tied Paragraphs

vtietoppar

After breaking a paragraph into lines, TEX computes the interline penalties by adding the values of: `\clubpenalty` after the first line of a paragraph.[3] ϵ-TEX generalizes the concept of interline, club, widow, and display widow penalty by allowing their replacement by arrays of penalty values.

vtiebotpar

After breaking a paragraph into lines, TEX computes the interline penalties by adding the values of: `\widowpenalty` before the last line of the paragraph. A float! ϵ-TEX generalizes the concept of interline, club, widow, and display widow penalty by allowing their replacement by arrays of penalty values.

vtiebotdisp

After breaking a paragraph into lines, TEX computes the interline penalties by adding the values of: `\displaywidowpenalty` before the line immediately preceding a displayed equation. ϵ-TEX generalizes the concept of interline, club, widow, and display widow penalty by allowing their replacement by arrays of penalty values.

$$gH = Hg \quad \text{for all} \quad g \in G.$$

Follow-up paragraph after and outside of the `vtiebotdisp`-environment.

vtiebotdisptoppar

After breaking a paragraph into lines, TEX computes the interline penalties by adding the values of: `\displaywidowpenalty` before the line immediately preceding a displayed equation. ϵ-TEX generalizes the concept of interline, club, widow, and display widow penalty by allowing their replacement by arrays of penalty values.

$$gH = Hg \quad \text{for all} \quad g \in G.$$

In this example we need a paragraph that follows the displayed math. So, we have to type some more text here to be able to demonstrate the action of the environment.

[3] Footnote of `vtietoppar`.

15 Breakable Displayed Equations

$$\begin{aligned}\overline{\psi}(x)\,\partial_\mu\,\psi(x) \mapsto \overline{\psi'}(x)\,\partial_\mu\,\psi'(x) &= e^{i\alpha(x)}\overline{\psi}(x)\,\partial_\mu\big(e^{-i\alpha(x)}\psi(x)\big)\\ &= \underbrace{\overline{\psi}(x)\,\partial_\mu\,\psi(x)}_{\text{free particle}} - i\,\overline{\psi}(x)\,\underbrace{\partial_\mu\big(\alpha(x)\big)}_{\text{vector field}}\,\psi(x).\end{aligned}$$

16 Setspace Front-End

Current settings are 10.6 pt/13 pt and `\typogfontsize` is 10.63995pt.

`\setbaselineskip{12pt plus 1pt minus .5pt}` New settings: 10.6 pt/12 pt.

When you are typsetting a document that spans several pages, it's generally best to define `\baselineskip` so that it cannot stretch or shrink, because this will give more uniformity to the pages. A small variation in the distance between the baselines—say only half a point—can make a substantial difference in the appearance of the type, since it significantly affects the proportion of white to black. On the other hand, if you are preparing a one-page document, you might want to give the baselineskip some stretchability, so that TEX will help you fit the copy on the page.

`\setbaselineskippercentage{130}` New settings: 10.6 pt/13.8 pt.

When you are typsetting a document that spans several pages, it's generally best to define `\baselineskip` so that it cannot stretch or shrink, because this will give more uniformity to the pages. A small variation in the distance between the baselines—say only half a point—can make a substantial difference in the appearance of the type, since it significantly affects the proportion of white to black. On the other hand, if you are preparing a one-page document, you might want to give the baselineskip some stretchability, so that TEX will help you fit the copy on the page.

`\setleading{1.5pt}` .0 New settings: 10.6 pt/12.1 pt.

When you are typsetting a document that spans several pages, it's generally best to define `\baselineskip` so that it cannot stretch or shrink, because this will give more uniformity to the pages. A small variation in the distance between the baselines—say only half a point—can make a substantial difference in the appearance of the type, since it significantly affects the proportion of white to black. On the other hand, if you are preparing a one-page document, you might want to give the baselineskip some stretchability, so that TEX will help you fit the copy on the page.

`\setleadingpercentage{30}` New settings: 10.6 pt/13.8 pt.

When you are typsetting a document that spans several pages, it's generally best to define `\baselineskip` so that it cannot stretch or shrink, because this will give more uniformity to the pages. A small variation in the distance between the baselines—say only half a point—can make a substantial difference in the appearance of the type, since it significantly affects the proportion of white to black. On the other hand, if you are preparing a one-page document, you might want to give the baselineskip some stretchability, so that TEX will help you fit the copy on the page.

The sample text was taken from The TEXbook [1, p. 78].

17 Smooth Ragged

Example 17.1 – Comparison of ragged right typesetting. The first example uses `RaggedRight` of ragged2e the second `smoothraggedrightpar` of typog. Both examples share a `\fussy` setting and a 10pt wide ragged right margin.

If you want to avoid overfull boxes at all costs without trying to fix them manually, you might be tempted to set `tolerance=10000`; this allows arbitrarily bad lines to be acceptable in tough situations. But infinite tolerance is a bad idea, because TEX doesn't distinguish between terribly bad and preposterously horrible lines. Indeed, a tolerance of 10000 encourages TEX to concentrate all the badness in one place, making one truly unsightly line instead of two moderately bad ones, because a single "write-off" produces fewest total demerits according to the rules.

If you want to avoid overfull boxes at all costs without trying to fix them manually, you might be tempted to set `tolerance=10000`; this allows arbitrarily bad lines to be acceptable in tough situations. But infinite tolerance is a bad idea, because TEX doesn't distinguish between terribly bad and preposterously horrible lines. Indeed, a tolerance of 10000 encourages TEX to concentrate all the badness in one place, making one truly unsightly line instead of two moderately bad ones, because a single "write-off" produces fewest total demerits according to the rules.

The sample text was taken from The TEXbook [1, p. 101].

The font used in this example is Inter regular, 9/11.

`\parindent`=15.0pt, visually: | |;
`\smoothraggedrightleftskip`=0.0pt. `\smoothraggedrightparindent`=0.0pt.

If you want to avoid overfull boxes at all costs without trying to fix them manually, you might be tempted to set `tolerance=10000`; this allows arbitrarily bad lines to be acceptable in tough situations. But infinite tolerance is a bad idea, because TEX doesn't distinguish between terribly bad and preposterously horrible lines. Indeed, a tolerance of 10000 encourages TEX to concentrate all the badness in one place, making one truly unsightly line instead of two moderately bad ones, because a single "write-off" produces fewest total demerits according to the rules.

If you want to avoid overfull boxes at all costs without trying to fix them manually, you might be tempted to set `tolerance=10000`; this allows arbitrarily bad lines to be acceptable in tough situations. But infinite tolerance is a bad idea, because TEX doesn't distinguish between terribly bad and preposterously horrible lines. Indeed, a tolerance of 10000 encourages TEX to concentrate all the badness in one place, making one truly unsightly line instead of two moderately bad ones, because a single "write-off" produces fewest total demerits according to the rules.

Eine Abbildung oder ein Operator A heißen nilpotent vom Grad k, falls $k \in N$ die kleinste Zahl ist, für die gilt: $A^k = 0$.

Eine Abbildung oder ein Operator A heißen nilpotent vom Grad k, falls $k \in N$ die kleinste Zahl ist, für die gilt: $A^k = 0$.

References

[1] KNUTH, D. E., *The TEXbook*, Vol. A of Computers&Typesetting, Addison Wesley, Reading/MA, 1986.