

1           *lineno.sty*   v5.9 2026/03/08

2

3                   A L<sup>A</sup>T<sub>E</sub>X package to attach  
4                   line numbers to paragraphs

5                               Stephan I. Böttcher  
                                  Uwe Lück  
                                  Karl Wette

6                   <https://github.com/latex-lineno/lineno>

7   **Contents**

8	<b>1</b>	<b>Introductions</b>	<b>2</b>
9	1.1	Introduction to versions v < 4 . . . . .	3
10	1.2	Introduction to versions v4.00ff. (UL) . . . . .	3
11	1.3	Availability . . . . .	5
12	1.4	Introductory code . . . . .	5
13	<b>2</b>	<b>Put the line numbers to the lines</b>	<b>6</b>
14	2.1	Basic code of <code>lineno.sty \output</code> . . . . .	6
15	2.2	<code>\LineNoTest</code> . . . . .	9
16	2.3	Other output routines (v4.2) . . . . .	9
17	2.4	<code>\MakeLineNo</code> : Actually attach line number . . . . .	11
18	<b>3</b>	<b>Control line numbering</b>	<b>14</b>
19	3.1	Inserting <code>\output</code> calls . . . . .	14
20	3.2	Turning on/off . . . . .	16
21	3.3	Display math . . . . .	18
22	<b>4</b>	<b>Line number references</b>	<b>20</b>
23	4.1	Internals . . . . .	20
24	4.2	The <code>\linelabel</code> command . . . . .	22

1	<b>5 The appearance of the line numbers</b>	<b>24</b>
2	5.1 Basic code . . . . .	24
3	5.2 Running line numbers . . . . .	27
4	5.3 Pagewise line numbers . . . . .	27
5	5.4 Twocolumn mode (New v3.06) . . . . .	32
6	5.5 Numbering modulo $m$ , starting at $f$ . . . . .	33
7	<b>6 Package options</b>	<b>37</b>
8	6.1 Extended referencing to line numbers. (v4.2) . . . . .	38
9	6.2 <code>\linelabel</code> in math mode . . . . .	38
10	6.3 Arrays, tabular environments (Revised v4.11) . . . . .	39
11	6.4 Switch among settings . . . . .	40
12	6.5 Compatibility with <code>hyperref</code> . . . . .	42
13	6.6 A note on calling so many options . . . . .	43
14	6.7 Execute options . . . . .	44
15	<b>7 Package extensions</b>	<b>44</b>
16	7.1 <code>displaymath</code> . . . . .	44
17	7.2 Line numbers in internal vertical mode . . . . .	45
18	7.3 Line number references with offset . . . . .	46
19	7.4 Numbered quotation environments . . . . .	47
20	7.5 Frame around a paragraph . . . . .	48
21	7.6 <code>amsmath</code> patches . . . . .	49
22	<b>8 Move <code>\vadjust</code> items (New v4.00)</b>	<b>51</b>
23	8.1 Redefining <code>\vadjust</code> . . . . .	51
24	8.2 Redefining the L <sup>A</sup> T <sub>E</sub> X commands . . . . .	52
25	8.3 Reminder on obsolescence . . . . .	54
26	<b>9 The final touch</b>	<b>54</b>
27	<b>10 The user commands</b>	<b>55</b>
28	10.1 Customization hooks . . . . .	58

## 29 1 Introductions

30 (New v4.00) Parts of former first section have been rendered separate sub-  
 31 sections for package version v4.00. (/New v4.00)

## 1.1 Introduction to versions $v < 4$

This package provides line numbers on paragraphs. After  $\text{T}_{\text{E}}\text{X}$  has broken a paragraph into lines there will be line numbers attached to them, with the possibility to make references through the  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$   $\backslash\text{ref}$ ,  $\backslash\text{pageref}$  cross reference mechanism. This includes four issues:

- attach a line number on each line,
- create references to a line number,
- control line numbering mode,
- count the lines and print the numbers.

The first two points are implemented through patches to the output routine. The third by redefining  $\backslash\text{par}$ ,  $\backslash\text{@par}$  and  $\backslash\text{@@par}$ . The counting is easy, as long as you want the line numbers run through the text. If they shall start over at the top of each page, the aux-file as well as  $\text{T}_{\text{E}}\text{X}$ 's memory have to carry a load for each counted line.

I wrote this package for my wife Petra, who needs it for transcriptions of interviews. This allows her to precisely refer to passages in the text. It works well together with  $\backslash\text{marginpars}$ , but not too well with  $\text{displaymath}$ .  $\backslash\text{footnotes}$  are a problem, especially when they are split, but we may get there. (New v4.00 UL) Version v4.00 overcomes the problem, I believe. (/UL /New v4.00)

`lineno.sty` works surprisingly well with other packages, for example, `wrapfig.sty`. So please try if it works with whatever you need, and if it does, please tell me, and if it does not, tell me as well, so I can try to fix it.

## 1.2 Introduction to versions v4.00ff. (UL)

`lineno.sty` has been maintained by Stephan until version v3.14. From version v4.00 onwards, maintenance is shifting towards Uwe Lück (UL), who is the author of v4...code and of v4...changes in documentation. This came about as follows.

Since late 2002, Christian Tapp and Uwe Lück have employed `lineno.sty` for their `ednotes.sty`, a package supporting critical editions, while you find `ednotes.sty` and surrounding files in CTAN folder `/macros/latex/contrib/ednotes`.

Soon, some weaknesses of `lineno.sty` showed up, mainly since Christian's critical editions (using `ednotes.sty`) needed lots of  $\backslash\text{linelabels}$  and footnotes. (These weaknesses are due to weaknesses of  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 's  $\backslash\text{marginpar}$

1 mechanism that Stephan used for `\linelabel`.) So we changed some  
2 `lineno.sty` definitions in some extra files, which moreover offered new fea-  
3 tures. We sent these files to Stephan, hoping he would take the changes into  
4 `lineno.sty`. However, he was too short of time.

5 Writing a TUGboat article on Ednotes in 2004, we hoped to reduce the  
6 number of files in the Ednotes bundle and so asked Stephan again. Now he  
7 generously offered maintenance to me, so I could execute the changes on my  
8 own.

9 The improvements are as follows:

10 (i) Footnotes placement approaches intentions better (footnotes formerly  
11 liked to pile up at late pages).

12 (ii) The number of `\linelabels` in one paragraph is no longer limited to  
13 18.

14 (iii) `\pagebreak`, `\nopagebreak`, `\vspace`, and the star and optional ver-  
15 sions of `\` work as one would expect (section 8).

16 (iv) A command is offered which chooses the first line number to be printed  
17 in the margin (subsection 5.5).

18 (v) (New v4.1)  $\text{\LaTeX}$  tabular environments (optionally) get line numbers  
19 as well, and you can refer to them in the usual automatic way. (It may  
20 be considered a shortcoming that, precisely, *rows* are numbered, not  
21 lines.—See subsection 6.3.)

22 (vi) We are moving towards referring to math items (subsection 6.2 and the  
23 hooks in subsection 4.2). (/New v4.1)

24 (Thanks to Stephan for making this possible!)

25 Ednotes moreover profits from Stephan's offer with regard to the doc-  
26 umentation of our code which yielded these improvements formerly. This  
27 documentation now becomes printable, being part of the `lineno.sty` docu-  
28 mentation.

29 Of course, Stephan's previous `lineno.sty` versions were a great and  
30 ingenious work and exhibit greatest  $\text{\TeX}$  expertise. I never could have done  
31 this. I learnt a lot in studying the code when Christian pointed out strange  
32 output results and error messages, and there are still large portions of  
33 `lineno.sty` which I don't understand (consider only pagewise numbering  
34 of lines). Fortunately, Stephan has offered future help if needed.—My code  
35 for attaching line numbers to *tabular environments* (as mentioned above,

1 now still in `edtable.sty`) developed from macros which Stephan and Chris-  
 2 tian experimented with in December 2002. Stephan built the basics. (How-  
 3 ever, I then became too proud to follow his advice only to use and modify  
 4 `longtable.sty`.)

5 There are some issues concerning use of counters on which I don't agree  
 6 with Stephan and where I would like to change the code if `lineno.sty` is  
 7 "mine" as Stephan offered. However, Stephan is afraid of compatibility prob-  
 8 lems from which, in particular, his wife could suffer in the near future. So he  
 9 demanded that I change as little as possible for my first version. Instead of  
 10 executing changes that I plan I just offer my opinions at the single occasions.  
 11 I hope to get in touch this way with users who consider subtle features vital  
 12 which I consider strange.

13 On the other hand, the sections on improvements of the implementation  
 14 have been blown up very much and may be tiring and litte understandable  
 15 for mere *users*. These users may profit from the present presentation just by  
 16 jumping to sections 6 and 10. There is a user's guide `ulinen.tex` which may  
 17 be even more helpful, but it has not been updated for a while.

### 18 1.3 Availability

19 In case you have found the present file otherwise than from CTAN: A recent  
 20 version and documentation of this package should be available from CTAN  
 21 folder `/macros/latex/contrib/lineno`. Or mail to one of the addresses at top  
 22 of file.

### 23 1.4 Introductory code

24 This style option is written for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>, November 1994 or later, since we  
 25 need the `\protected@write` macro.

26 (New v4.00) And we use `\newcommand*` for controlling length of user  
 27 macro arguments, which has been available since December 1994. (/New  
 28 v4.00)

```

1 \NeedsTeXFormat{LaTeX2e}[1994/12/01]
2 \ProvidesPackage{lineno}
3 [\filedate\space line numbers on paragraphs \fileversion]
4 \RequirePackage{etoolbox}
5 \RequirePackage{kvoptions}

```

## 1 2 Put the line numbers to the lines

2 (New v4.00) This section contained the most basic package code previously.  
 3 For various purposes of version 4... , much of these basics have been to be  
 4 modified. Much of my (UL's) reasoning on these modifications has been  
 5 to be reported. Sorry, the present section has been blown up awfully thus  
 6 and contains ramifications that may be difficult to trace. We add some  
 7 `\subsection` commands in order to cope with the new situation. (/New  
 8 v4.00)

### 9 2.1 Basic code of `lineno.sty` \output

10 The line numbers have to be attached by the output routine. We simply set  
 11 the `\interlinepenalty` to  $-100000$ . The output routine will be called after  
 12 each line in the paragraph, except the last, where we trigger by `\par`. The  
 13 `\linenopenalty` is small enough to compensate a bunch of penalties (e.g.,  
 14 with `\samepage`).

15 (New v3.04) Longtable uses `\penalty-30000`. The `lineno` penalty range  
 16 was shrunk to  $-188000 \dots -32000$ . (/New v3.04) (New v4.00) New values  
 17 are listed below (11111f.). (/New v4.00)

```
6 \newcount\linenopenalty\linenopenalty=-100000
```

18 (UL) Hm. It is never needed below that this is a counter.  
 19 `\def\linenopenalty{-100000\relax}` would do. (I guess this consumes  
 20 more memory, but it is more important to save counters than to save mem-  
 21 ory.) I was frightened by `-\linenopenalty` below, but indeed  $\TeX$  interprets  
 22 the string `--100000` as `100000`. Has any user or extension package writer ever  
 23 called `\linenopenalty=xxx`, or could I really change this?—The counter is  
 24 somewhat faster than the macro. Together with the compatibility question  
 25 this seems to support keeping the counter. (???) (/UL)

```
7 \mathchardef\linenopenaltypar=32000
```

26 So let's make a hook to `\output`, the direct way. The  $\LaTeX$  macro  
 27 `\@reinserts` puts the footnotes back on the page.

28 (New v3.01) `\@reinserts` badly screws up split footnotes. The bottom  
 29 part is still on the recent contributions list, and the top part will be put back  
 30 there after the bottom part. Thus, since `lineno.sty` does not play well with  
 31 `\inserts` anyway, we can safely experiment with `\holdinginserts`, without  
 32 making things much worse.

1 Or that's what I thought, but: Just activating `\holdinginserts` while  
 2 doing the `\par` will not do the trick: The `\output` routine may be called  
 3 for a real page break before all line numbers are done, and how can we get  
 4 control over `\holdinginserts` at that point?

5 Let's try this: When the `\output` routine is run with `\holdinginserts=3`  
 6 for a real page break, then we reset `\holdinginserts` and restart `\output`.

7 Then, again, how do we keep the remaining `\inserts` while doing further  
 8 line numbers?

9 If we find `\holdinginserts=-3` we activate it again after doing `\output`.  
 10 (/New v3.01)

11 (New v3.02) To work with `multicol.sty`, the original output routine is  
 12 now called indirectly, instead of being replaced. When `multicol.sty` changes  
 13 `\output`, it is a toks register, not the real thing. (/New v3.02)

14 (New v4.00) Two further complications are added.

15 (i) Problems with footnotes formerly resulted from L<sup>A</sup>T<sub>E</sub>X's `\@reinserts`  
 16 in `\@specialoutput` which Stephan's `\linelabel` called via the  
 17 `\marginpar` mechanism.

18 (ii) L<sup>A</sup>T<sub>E</sub>X commands using `\vadjust` formerly didn't work as one would  
 19 have hoped. The problem is as follows: Printing the line num-  
 20 ber results from a box that the output routine inserts at the  
 21 place of the `\interlinepenalty`. `\vadjust` items appear *above* the  
 22 `\interlinepenalty` (T<sub>E</sub>Xbook p. 105). So `\pagebreak`, e.g., for-  
 23 merly sent the line number to the next page, while the penalty from  
 24 `\nopagebreak` could not tie the following line, since it was screened  
 25 off by the line number box.—Our trick is putting the `\vadjust` items  
 26 into a list macro from which the output routine transfers them into the  
 27 vertical list, below the line number box.

28 In this case (ii), like in case (i), footnotes would suffer if `\holdinginserts`  
 29 were non-positive. Indeed, in both cases (i) and (ii) we tackle the foot-  
 30 note problem by extending that part of Stephan's output routine that  
 31 is active when `\holdinginserts` is positive. This extension writes the  
 32 line number `\newlabel` to the `.aux` file (which was formerly done under  
 33 `\holdinginserts = -3`) and handles the `\vadjust` items.—To trigger  
 34 `\output` and its `\linelabel` or, resp., `\vadjust` part, the list of signal penal-  
 35 ties started immediately before is increased here (first for `\linelabel`, second  
 36 for postponed `\vadjust` items):

8 `\mathchardef\@Mllbcodepen=11111`

9 `\mathchardef\@Mppvacodepen=11112`

1 (/New v4.00) (New v4.2) David Kastrup urges to use a private name instead  
 2 of `\the\output` (LaTeX-L-list). Otherwise an `\output` routine loaded later  
 3 and using `\newtoks\output` again may get lost entirely. So we change use of  
 4 `\@LN@output`, using it for the former purpose. Reference to what appeared  
 5 with the name of `\output` here lasts for a few lines and then is given away.

```
10 \let\@tempa\output
11 \newtoks\output
12 \let\@LN@output\output
13 \output=\expandafter{\the\@tempa}
```

6 Now we add two cases to Stephan's output routine. (New v4.00)

```
14 \@tempa={%
```

7 (/New 4.2)

```
15         \LineNoTest
16         \if@tempswa
```

8 (New v4.00) We insert recognition of waiting `\linelabel` items—

```
17         \ifnum\outputpenalty=-\@Mllbcodepen
18         \WriteLineNo
```

9 —and of waiting `\vadjust` items:

```
19         \else
20         \ifnum\outputpenalty=-\@Mppvacodepen
21         \PassVadjustList
22         \else
```

10 (/New v4.00) (New v4.2) Outsource “Standard” output—which occurs so  
 11 rarely—to subsection 2.3:

```
23         \LineNoLaTeXOutput
```

12 (/New v4.2) (New v4.00) Two new `\fis` for the `\linelabel` and `\vadjust`  
 13 tests—

```
24         \fi
25         \fi
```

14 —and the remaining is Stephan's code again: (/New v4.00)

```
26         \else
27         \MakeLineNo
28         \fi
29         }
```

15 (New v4.00) Our new macros `\WriteLineNo` and `\PassVadjustList` will be  
 16 dealt with in sections 4 and 8.1. (/New v4.00)

## 1 2.2 \LineNoTest

2 The float mechanism inserts `\interlinepenalty`s during `\output`. So care-  
 3 fully reset it before going on. Else we get doubled line numbers on every  
 4 float placed in horizontal mode, e.g, from `\linelabel`.

5 Sorry, neither a `\linelabel` nor a `\marginpar` should insert a penalty,  
 6 else the following line number could go to the next page. Nor should any  
 7 other float. So let us suppress the `\interlinepenalty` altogether with the  
 8 `\@nbreak` switch.

9 Since (ltspace.dtx, v1.2p)[1996/07/26], the `\@nbreaktrue` does it's job  
 10 globally. We need to do it locally here.

```

30 \def\LineNoTest{%
31   \let\@par\@@par
32   \ifnum\interlinepenalty<-\linenopenaltypar
33     \advance\interlinepenalty-\linenopenalty
34     \@LN@nbreaktrue
35     \fi
36   \@tempswatru
37   \ifnum\outputpenalty>-\linenopenaltypar\else
38     \ifnum\outputpenalty>-188000\relax
39       \@tempswafalse
40       \fi
41     \fi
42   }
43
44 \def\@LN@nbreaktrue{\let\if@nbreak\iftrue} % renamed v4.33

```

11 (UL) I thought here were another case of the save stack problem ex-  
 12 plained in *TEXbook*, p. 301, namely through both local and global chang-  
 13 ing `\if@nbreak`. However, `\@LN@nbreak` is called during `\@LN@output`  
 14 only, while `\@nbreaktrue` is called by *L*<sup>A</sup>*T*<sub>E</sub>*X*'s `\@startsection` only.  
 15 The latter never happens during `\@LN@output`. So there is no local  
 16 value of `\if@nbreak` on save stack when `\@nbreaktrue` acts, since  
 17 `\the\@LN@output` (where `\@LN@output` is a new name for the original  
 18 `\output`) is executed within a group (*TEXbook* p. 21). (/UL)

## 19 2.3 Other output routines (v4.2)

20 I had thought of dealing with bad interference of footnotes (and  
 21 `\enlargethispage`) with (real) `\marginpars` and floats *here*. Yet this is  
 22 done in

[http://\[CTAN\]/macros/latex/contrib/tamefloats/tameflts.sty](http://[CTAN]/macros/latex/contrib/tamefloats/tameflts.sty)

1 now, and I prefer striving for compatibility with the latter. (See there for ex-  
 2 panding on the problem.) This requires returning the special absolute value  
 3 of `\holdinginserts` that `lineno.sty` finds at the end of a newly type-  
 4 set paragraph—now done in subsection 3.1 (`\linenumberpar`). The former  
 5 `\LineNoHoldInsertsTest` has been filled into here. Note: when the follow-  
 6 ing code is invoked, we have `\if@tempswa = \iftrue`. WARNING: I am  
 7 still not sure whether the present code is good for cooperating with other  
 8 packages that use `\holdinginserts`.

```

45 \def\LineNoLaTeXOutput{%
46   \ifnum \holdinginserts=\thr@@ % v4.33 without \@tempswafalse
47     \global\holdinginserts=\thr@@
48     \unvbox\@cclv
49     \ifnum \outputpenalty=\@M \else \penalty\outputpenalty \fi
50   \else
51     \if@twocolumn \let\@makecol\@LN@makecol \fi
52     \the\@LN@output % finally following David Kastrup's advice.
53     \ifnum \holdinginserts=-\thr@@
54       \global\holdinginserts=\thr@@ \fi
55   \fi
56 }

```

9 *More on dealing with output routines from other packages:* Since  
 10 `lineno.sty`'s output routine is called at least once for each output line,  
 11 I think it should be in  $\text{\TeX}$ 's original `\output`, while output routines deal-  
 12 ing with building pages and with floats etc. should be filled into registers  
 13 addressed by `\output` after `\newtoks\output`. Therefore

- 14 1. `tameflts.sty` should be loaded *after* `lineno.sty`;
- 15 2. if a class changes `\output` (APS journal class `revtex4`, e.g.),  
 16 `lineno.sty` should be loaded by `\RequirePackage` [here pre-  
 17 sumably following some options in brackets]{`lineno`} *preceding*  
 18 `\documentclass`.
- 19 3. If you actually maintain such a class, please consider loading  
 20 `lineno.sty` on some draft option. The bunch of `lineno`'s package op-  
 21 tions may be a problem, but perhaps the purpose of your class is offering  
 22 only very few of `lineno`'s options anyway, maybe just one.

23 The latter may also be needed with classes that don't follow David Kastrup's  
 24 rule on changing `\output`.

## 1 2.4 \MakeLineNo: Actually attach line number

2 We have to return all the page to the current page, and add a box with the  
 3 line number, without adding breakpoints, glue or space. The depth of our  
 4 line number should be equal to the previous depth of the page, in case the  
 5 page breaks here, and the box has to be moved up by that depth.

6 The `\interlinepenalty` comes after the `\vadjust` from a `\linelabel`,  
 7 so we increment the line number *after* printing it. The macro  
 8 `\makeLineNumber` produces the text of the line number, see section 5.

9 (UL) I needed a while to understand the sentence on incrementing. Cor-  
 10 rectly: writing the `\newlabel` to the `.aux` file is triggered by the signal  
 11 penalty that `\end@float` inserts via `\vadjust`. However, this could be  
 12 changed by our new `\PostponeVadjust`. After `\c@linenumber` has been in-  
 13 troduced as a L<sup>A</sup>T<sub>E</sub>X counter, it might be preferable that it behaved like stan-  
 14 dard L<sup>A</sup>T<sub>E</sub>X counters which are incremented shortly before printing. But this  
 15 may be of little practical relevance in this case, as `\c@linenumber` is driven in  
 16 a very non-standard way.—However still, this behaviour of `\c@linenumber`  
 17 generates a problem with our `edtable.sty`. (/UL).

18 Finally we put in the natural `\interlinepenalty`, except after the last  
 19 line.

20 (New v3.10) Frank Mittelbach points out that `box255` may be less deep  
 21 than the last box inside, so he proposes to measure the page depth with  
 22 `\boxmaxdepth=\maxdimen`. (/New v3.10)

23 (UL, New v4.00) We also resume the matter of `\vadjust` items that was  
 24 started in section 2.1.

25 T<sub>E</sub>X puts only nonzero interline penalties into the vertical list (T<sub>E</sub>Xbook  
 26 p. 105), while `lineno.sty` formerly replaced the signal interline penalty by  
 27 something closing with an explicit penalty of the value that the interline  
 28 penalty would have without `lineno.sty`. This is usually 0. Now, ex-  
 29 plicit vertical penalties can be very nasty with respect to `\nopagebreak`,  
 30 e.g., a low (even positive) `\widowpenalty` may force a widow where you  
 31 explicitly tried to forbid it by `\nopagebreak` (see explanation soon below).  
 32 The `\nopagebreak` we create here would never work if all those zero penal-  
 33 ties were present.—On the other hand, we cannot just omit Stephan's zero  
 34 penalties, because T<sub>E</sub>X puts a penalty of 10000 after what `lineno.sty` in-  
 35 serts (T<sub>E</sub>Xbook p. 125). This penalty must be overridden to allow page  
 36 breaks between ordinary lines. To revive `\nopagebreak`, we therefore re-  
 37 place those zero (or low) penalties by penalties that the user demanded by  
 38 `\nopagebreak`.—This mechanism is not perfect and does not exactly restore  
 39 the original L<sup>A</sup>T<sub>E</sub>X working of `\pagebreak` and `\nopagebreak`. Viz., if there  
 40 are several vertical penalties after a line which were produced by closely

1 sitting `\[no]pagebreaks`, without `lineno.sty` the lowest penalty would be  
 2 effective (cf. `TeXbook` exercise 14.10). Our mechanism, by contrast, chooses  
 3 the *last* user-set penalty of the line as the effective one. It would not be very  
 4 difficult to come more close to the original mechanism, but until someone  
 5 urges us we will cling to the present simple way. You may consider an ad-  
 6 vantage of the difference between our mechanism and the original one that  
 7 the user here can actually override low penalties by `\nopagebreak`, which  
 8 may be what a lay L<sup>A</sup>T<sub>E</sub>X user would expect. (/UL, /New v4.00)

```
57 \def\MakeLineNo{%
58   \@LN@maybe@normalLineNumber           % v4.31
59   \boxmaxdepth\maxdimen\setbox\z@\vbox{\unvbox\@cclv}%
60   \@tempdima\dp\z@ \unvbox\z@
61   \sbox\@tempboxa{\hb@xt@\z@{\makeLineNumber}}%
```

9 (New v4.00) Previously,

```
10 %   \stepcounter{linenumber}%
```

11 followed. (Of course, there was no comment mark; I put it there to make  
 12 reading the actual code easy.)

13 (New v4.22: improved) Why not just

```
\global\advance\c@linenumber\@ne?
```

14 `\stepcounter` additionally resets “subordinate” counters, but which could  
 15 these (usefully) be? Again, may be column counters with `edtable.sty`!

16 But then, our `edtable.sty` and its `longtable` option should use it as  
 17 well. So use a shorthand supporting uniformity. You can even use it as  
 18 a hook for choosing `\global\advance\c@linenumber\@ne` instead of our  
 19 choice. (/New v4.22)

```
62 \stepLineNumber
```

20 (New v4.4) Now

```
63 \ht\@tempboxa\z@ \@LN@depthbox
```

21 appends the box containing the line number without changing `\prevdepth`—  
 22 see end of section. Now is the time for inserting the ... (/New v4.4) `\adjust`  
 23 items. We cannot do this much later, because their right place is above the  
 24 artificial interline penalty which Stephan’s code will soon insert (cf. `TeXbook`  
 25 p. 105). The next command is just `\relax` if no `\adjust` items have been  
 26 accumulated for the current line. Otherwise it is a list macro inserting the  
 27 `\adjust` items and finally resetting itself. (This is made in section 8.1  
 28 below.) If the final item is a penalty, it is stored so it can compete with  
 29 other things about page breaking.

```
64 \LN@do@vadjusts
65 \count@\lastpenalty
```

1 At this place,

```
2 % \ifnum\outputpenalty=-\linenopenaltypar\else
```

3 originally followed. We need something *before* the `\else`:

```
66 \ifnum\outputpenalty=-\linenopenaltypar
67 \ifnum\count@=\z@ \else
```

4 So final `\pagebreak[0]` or `\nopagebreak[0]` has no effect—but this will  
5 make a difference after headings only, where nobody should place such a  
6 thing anyway.

```
68 \xdef\LN@parpgbrk{%
69 \penalty\the\count@
70 \global\let\noexpand\LN@parpgbrk
71 \noexpand\LN@screenoff@pen}% v4.4
```

7 That penalty will replace former `\kern\z@` in `\linenumberpar`, see subsec-  
8 tion 3.1.—A few days earlier, I tried to send just a penalty value. However,  
9 the `\kern\z@` in `\linenumberpar` is crucial, as I then found out. See below.—  
10 The final penalty is repeated, but this does no harm. (It would not be very  
11 difficult to avoid the repeating, but it may even be less efficient.) It may be  
12 repeated due to the previous `\xdef`, but it may be repeated as well below in  
13 the present macro where artificial interline penalty is to be overridden.

```
72 \fi
73 \else
```

14 (/New v4.00)

```
74 \@tempcnta\outputpenalty
75 \advance\@tempcnta -\linenopenalty
```

15 (New v4.00)

```
16 % \penalty\@tempcnta
```

17 followed previously. To give `\nopagebreak` a chance, we do

```
76 \penalty \ifnum\count@<\@tempcnta \@tempcnta \else \count@ \fi
```

1 instead.—In `linenox0.sty`, the `\else` thing once was omitted. Sergei  
 2 Mariev’s complaint (thanks!) showed that it is vital (see comment before  
 3 `\MakeLineNo`). The remaining `\fi` from previous package version closes the  
 4 `\ifnum\outputpenalty...` (/New v4.00)

```
77 \fi
78 }
```

5 (New v4.00)

```
79 \newcommand\stepLineNumber{\stepcounter{linenumber}}
```

6 For reason, see use above. (/New v4.00)  
 7 (New v4.4) The depth preserving trick is drawn here from `\MakeLineNo`  
 8 because it will be used again in section 3.1. (v5.3) Handle special value of  
 9 `\prevdepth=-1000pt`. (/v5.3)

```
80 \def\LN@depthbox{%
81 \ifdim\@tempdima=-1000pt
82 % \nointerlineskip is already set so we don't need set it again
83 % (and we shouldn't back up)
84 \else
85 \dp\@tempboxa=\@tempdima
86 \nointerlineskip
87 \kern-\@tempdima
88 \fi
89 \box\@tempboxa}
```

10 (/New v4.4)

## 11 3 Control line numbering

### 12 3.1 Inserting `\output` calls

13 The line numbering is controlled via `\par`. L<sup>A</sup>T<sub>E</sub>X saved the T<sub>E</sub>X-primitive  
 14 `\par` in `\@@par`. We push it one level further out, and redefine `\@@par` to  
 15 insert the `\interlinepenalty` needed to trigger the line numbering. And  
 16 we need to allow pagebreaks after a paragraph.

17 New (2.05beta): the `prevgraf` test. A paragraph that ends  
 18 with a displayed equation, a `\noindent\par` or `wrapfig.sty` produce  
 19 empty paragraphs. These should not get a spurious line number via  
 20 `\linenopenaltypar`.

```
90 \let\@@@par\@@par
91 \newcount\linenoprevgraf
```

1 (UL) And needs `\linenoprevgraf` to be a counter? Perhaps there may  
 2 be a paragraph having thousands of lines, so `\mathchardef` doesn't suffice  
 3 (really??). A macro ending on `\relax` might suffice, but would be somewhat  
 4 slow. I think I will use `\mathchardef` next time. Or has any user used  
 5 `\linenoprevgraf`? (/UL)

```

92 \def\linenumberpar{%
93   \ifvmode \@@@par \else
94     \ifinner \@@@par \else
95       \xdef\@LN@outer@holdins{\the\holdinginserts}% v4.2
96       \advance \interlinepenalty \linenopenalty
97       \linenoprevgraf \prevgraf
98       \global \holdinginserts \thr@@
99       \@@@par
100      \ifnum\prevgraf>\linenoprevgraf
101        \penalty-\linenopenaltypar
102      \fi

```

6 (New v4.00)

```

7 %           \kern\z@

```

8 was here previously. What for? According to *TEXbook* p. 125, Stephan's  
 9 interline penalty is changed into 10000. At the end of a paragraph, the  
 10 `\parskip` would follow that penalty of 10000, so there could be a page break  
 11 neither at the `\parskip` nor at the `\baselineskip` (*TEXbook* p. 110)—so  
 12 there could never be a page break between two paragraphs. So something  
 13 must screen off the 10000 penalty. Indeed, the `\kern` is a place to break.  
 14 (Stephan once knew this: see 'allow pagebreaks' above.)

15 Formerly, I tried to replace `\kern\z@` by

```

16 %           \penalty\@LN@parpgpen\relax

```

17 —but this allows a page break after heading. So:

```

103          \@LN@parpgbrk

```

18 These and similar changes were formerly done by `linenox1.sty`. (/New  
 19 v4.00)

20 (New v4.4) A `\belowdisplayskip` may precede the previous when the  
 21 paragraph ends on a display-math; or there may be a `\topsep` from a list, etc.  
 22 `\addvspace` couldn't take account for it with `\kern\z@` here. v4.32 therefore  
 23 moved the space down – with at least two bad consequences. Moreover, David  
 24 Josef Dev observes that `\kern\z@` may inappropriately yield column depth  
 25 0pt. For these reasons, we introduce `\@LN@screenoff@open` below. (/New  
 26 v4.4)

```

104     \global\holdinginserts\@LN@outer@holdins % v4.2
105     \advance\interlinepenalty -\linenopenalty
106     \fi      % from \ifinner ... \else
107     \fi}     % from \ifvmode ... \else

```

1 (New v4.00, v4.4) Initialize `\@LN@parpgbrk`, accounting for earlier space  
 2 and for appropriate columndepth. We use former `\MakeLineNo`'s depth-  
 3 preverving trick `\@LN@depthbox` again:

```

108 \def\@LN@screenoff@open{%
109   \ifdim\lastskip=\z@
110     \@tempdima\prevdepth \setbox\@tempboxa\@null
111     \@LN@depthbox          \fi}
112
113 \global\let\@LN@parpgbrk\@LN@screenoff@open

```

4 (/New v4.4, v4.00)

## 5 3.2 Turning on/off

6 The basic commands to enable and disable line numbers. `\@par` and `\par`  
 7 are only touched, when they are `\let` to `\@@@par/\linenumberpar`. The line  
 8 number may be reset to 1 with the star-form, or set by an optional argument  
 9 [*number*].

10 (New v4.00) We add `\ifLineNumbers` etc. since a number of our new ad-  
 11 justments need to know whether linenumbers is active. This just provides a  
 12 kind of shorthand for `\ifx\@@@par\linenumberpar`; moreover it is more sta-  
 13 ble: who knows what may happen to `\@@@par`?—A caveat: `\ifLineNumbers`  
 14 may be wrong. E.g., it may be `\iffalse` where it acts, while a `\linenumbers`  
 15 a few lines below—in the same paragraph—brings about that the line where  
 16 the `\ifLineNumbers` appears gets a marginal number. (New v4.3) Just  
 17 noticed: Such tricks have been disallowed with v4.11, see subsections 4.2  
 18 and 3.2.—Moreover, the switching between meanings of `\linelabel` for a  
 19 possible error message as of v4.11 is removed. Speed is difficult to esteem  
 20 and also depends on applications. Just use the most simple code you find.  
 21 (/New v4.3)

```

114 \newif\ifLineNumbers \LineNumbersfalse

```

22 (/New v4.00)

```

115 \def\linenumbers{%
116   \LineNumberstrue          % v4.00
117   \xdef\@LN@outer@holdins{\the\holdinginserts}% v4.3

```

1 (New v4.3) The previous line is for `{\linenomath}` in a first numbered para-  
 2 graph. (/New v4.3)

```

118 \let\@par\linenumberpar
119 % \let\linelabel\LN@linelabel % v4.11, removed v4.3
120 \ifx\@par\@@par\let\@par\linenumberpar\fi
121 \ifx\par\@@par\let\par\linenumberpar\fi
122 \@LN@maybe@moduloresume % v4.31
123 \@ifnextchar[{\resetlinenumber}%]
124 {\@ifstar{\resetlinenumber}{}}%
125 }
126
127 \def\nolinenumbers{%
128 \LineNumbersfalse % v4.00
129 \let\@par\@@par
130 % \let\linelabel\LN@LError % v4.11, removed v4.3
131 \ifx\@par\linenumberpar\let\@par\@@par\fi
132 \ifx\par\linenumberpar\let\par\@@par\fi
133 }

```

3 (New v4.00) Moreover, it is useful to switch to `\nolinenumbers` in  
 4 `\@arrayparboxrestore`. We postpone this to section 8.2 where we'll have  
 5 an appending macro for doing this. (/New v4.00)

6 What happens with a display math? Since `\par` is not executed, when  
 7 breaking the lines before a display, they will not get line numbers. Sorry,  
 8 but I do not dare to change `\interlinepenalty` globally, nor do I want to  
 9 redefine the display math environments here.

### *display math*

10 See the subsection below, for a wrapper environment to make it work. But  
 11 that requires to wrap each and every display in your  $\LaTeX$  source (see option  
 12 `displaymath` in subsections 6.4 and 7.1 for some relief [UL]).

13 The next two commands are provided to turn on line numbering in  
 14 a specific mode. Please note the difference: for pagewise numbering,  
 15 `\linenumbers` comes first to inhibit it from seeing optional arguments, since  
 16 re-/presetting the counter is useless.

```

134 \def\pagewiselinenumbers{\linenumbers\setpagewiselinenumbers}
135 \def\runninglinenumbers{\setrunninglinenumbers\linenumbers}

```

17 Finally, it is a  $\LaTeX$  style, so we provide for the use of environments, includ-  
 18 ing the suppression of the following paragraph's indentation.

19 (UL) I am drawing the following private thoughts of Stephan's to publicity  
 20 so that others may think about them—or to remind myself of them in an  
 21 efficient way. (/UL)

```

1 % TO DO: add \par to \linenumbers, if called from an environment.
2 % To DO: add an \@endpe hack if \linenumbers are turned on
3 %       in horizontal mode. {\par\parskip\z@\noindent} or
4 %       something.

5 (UL) However, I rather think that \linenumbers and \nolinenumbers
6 should execute a \par already. (Then the \pars in the following definitions
7 should be removed.) (/UL)

136 \@namedef{linenumbers*}{\par\linenumbers*}
137 \@namedef{runninglinenumbers*}{\par\runninglinenumbers*}
138
139 \def\endlinenumbers{\par\@endpetrue}
140 \let\endrunninglinenumbers\endlinenumbers
141 \let\endpagewiselinenumbers\endlinenumbers
142 \expandafter\let\csname endlinenumbers*\endcsname\endlinenumbers
143 \expandafter\let\csname endrunninglinenumbers*\endcsname\endlinenumbers
144 \let\endnolinenumbers\endlinenumbers

```

### 8 3.3 Display math

9 Now we tackle the problem to get display math working. There are different  
10 options.

- 11 1. Precede every display math with a \par. Not too good.
- 12 2. Change \interlinepenalty and associates globally. Unstable.
- 13 3. Wrap each display math with a `{\linenomath}` environment.

14 We'll go for option 3. See if it works:

$$display\ math \tag{1}$$

15 The star form `{\linenomath*}` should also number the lines of the display  
16 itself,

$$multi \qquad \qquad \qquad line \tag{2}$$

$$display \qquad \qquad \qquad math \tag{3}$$

$$with \tag{4}$$

$$array$$

20 including multiline displays.

21 First, here are two macros to turn on linenumbers on paragraphs pre-  
22 ceeding displays, with numbering the lines of the display itself, or without.

- 1 The `\ifx..` tests if line numbering is turned on. It does not harm to add  
 2 these wrappers in sections that are not numbered. Nor does it harm to wrap  
 3 a display twice, e.g, in case you have some `{equation}`s wrapped explicitly,  
 4 and later you redefine `\equation` to do it automatically.  
 5 (New v4.3) To avoid the spurious line number above a display in vmode,  
 6 I insert `\ifhmode.` (/New v4.3)

```

145 \newcommand\linenomathNonnumbers{%
146   \ifLineNumbers
147     \ifnum\interlinepenalty>-\linenopenaltypar
148       \global\holdinginserts\thr@@
149       \advance\interlinepenalty \linenopenalty
150       \ifhmode % v4.3
151         \advance\predisplaypenalty \linenopenalty
152       \fi
153     \fi
154   \fi
155   \ignorespaces
156 }
157
158 \newcommand\linenomathWithnumbers{%
159   \ifLineNumbers
160     \ifnum\interlinepenalty>-\linenopenaltypar
161       \global\holdinginserts\thr@@
162       \advance\interlinepenalty \linenopenalty
163       \ifhmode % v4.3
164         \advance\predisplaypenalty \linenopenalty
165       \fi
166       \advance\postdisplaypenalty \linenopenalty
167       \advance\interdisplaylinepenalty \linenopenalty
168     \fi
169   \fi
170   \ignorespaces
171 }

```

- 7 The `{linenomath}` environment has two forms, with and without a star. The  
 8 following two macros define the environment, where the starred/non-starred  
 9 form does/doesn't number the lines of the display or vice versa.

```

172 \newcommand\linenumberdisplaymath{%
173   \def\linenomath{\linenomathWithnumbers}%
174   \@namedef{linenomath*}{\linenomathNonnumbers}%
175 }
176
177 \newcommand\nolinenumberdisplaymath{%
178   \def\linenomath{\linenomathNonnumbers}%
179   \@namedef{linenomath*}{\linenomathWithnumbers}%

```

```

180 }
181
182 \def\endlinenomath{%
183   \ifLineNumbers                % v4.3
184   \global\holdinginserts\@LN@outer@holdins % v4.21
185   \fi
186   \global % v4.21 support for LaTeX2e earlier than 1996/07/26.
187   \@ignoretrue
188 }
189 \expandafter\let\csname endlinenomath*\endcsname\endlinenomath

```

1 The default is not to number the lines of a display. But the package option  
2 `mathlines` may be used to switch that behavior.

```

190 \nolinenumberdisplaymath

```

## 3 4 Line number references

### 4 4.1 Internals

5 The only way to get a label to a line number in a paragraph is to ask the  
6 output routine to mark it.

7 (New v4.00) The following two paragraphs don't hold any longer, see  
8 below. (/New v4.00)

```

9 % We use the marginpar mechanism to hook to ~\output~ for a
10 % second time. Marginpars are floats with number $-1$, we
11 % fake marginpars with No $-2$. Originally, every negative
12 % numbered float was considered to be a marginpar.
13 %
14 % The float box number ~\@currbox~ is used to transfer the
15 % label name in a macro called ~\@LNL@~\angle{box-number}.

```

16 A `\newlabel` is written to the aux-file. The reference is to `\theLineNumber`,  
17 *not* `\thelinenumber`. This allows to hook in, as done below for pagewise  
18 line numbering.

19 (New v3.03) The `\@LN@ExtraLabelItems` are added for a hook to keep  
20 packages like `{hyperref}` happy. (/New v3.03)

21 (New v4.00) We fire the `\marginpar` mechanism, so we leave L<sup>A</sup>T<sub>E</sub>X's  
22 `\@addmarginpar` untouched.

```

23 % \let\@LN@addmarginpar\@addmarginpar
24 % \def\@addmarginpar{%
25 %   \ifnum\count\@currbox>-2\relax
26 %     \expandafter\@LN@addmarginpar

```

```

1 % \else
2 %   \@cons\@freelist\@currbox
3 %   \protected@write\@auxout{}\@%
4 %     \string\newlabel
5 %       {\csname @LNL@the\@currbox\endcsname}%
6 %       {\theLineNumber}\thepage\@LN@ExtraLabelItems}}%
7 % \fi}

```

8 OK, we keep Stephan's \@LN@ExtraLabelItems: (/New v4.00)

```
191 \let\@LN@ExtraLabelItems\@empty
```

9 (New v4.00) We imitate the \marginpar mechanism without using the  
10 \@freelist boxes. \linelabel will indeed place a signal penalty  
11 (\@Mllbcodepen, new), and it will put a label into some list macro  
12 \@LN@labellist. A new part of the output routine will take the labels  
13 from the list and will write \newlabels to the .aux file.

14 The following is a version of L<sup>A</sup>T<sub>E</sub>X's \@xnext.

```
192 \def\@LN@xnext#1\@lt#2\@#3#4{\def#3{#1}\gdef#4{#2}}
```

15 This takes an item #1 from a list #4 into #3; to be used as  
16 \expandafter\@LN@xnext#4\@#3#4. Our lists use \@lt after each item  
17 for separating. Indeed, there will be another list macro which can appear as  
18 argument #4, this will be used for moving \vadjust items (section 8.1). The  
19 list for \linelabels is the following:

```
193 \global\let\@LN@labellist\@empty
```

20 The next is the new part of the output routine writing the \newlabel to the  
21 .aux file. Since it is no real page output, the page is put back to top of the  
22 main vertical list.

```

194 \def\WriteLineNo{%
195   \unvbox\@cclv
196   \expandafter \@LN@xnext \@LN@labellist \@
197     \@LN@label \@LN@labellist
198   \protected@write\@auxout{}\string\newlabel{\@LN@label}%
199     {\theLineNumber}\thepage\@LN@ExtraLabelItems}}%
200 }

```

23 (/New v4.00)

## 1 4.2 The `\linelabel` command

2 To refer to a place in line `\ref{\foo}` at page `\pageref{\foo}` you place a  
3 `\linelabel{\foo}` at that place.

4 (New v4.11)

```
5 % If you use this command outside a ~\linenumbers~
6 % paragraph, you will get references to some bogus
7 % line numbers, sorry. But we don't disable the command,
8 % because only the ~\par~ at the end of a paragraph may
9 % decide whether to print line numbers on this paragraph
10 % or not. A ~\linelabel~ may legally appear earlier than
11 % ~\linenumbers~.
```

See if it  
works:  
This  
paragraph  
starts on  
page 22,  
line 4.

12 This trick is better not allowed—see subsections 4.2 and 3.2. (/New v4.11)

13 `\linelabel`

14 `%`, via a fake float number `$$-2$`, `%%` new mechanism v4.00

15 puts a `\penalty` into a `\vadjust`, which triggers the pagebuilder after  
16 putting the current line to the main vertical list. A `\write` is placed  
17 on the main vertical list, which prints a reference to the current value of  
18 `\thelinenumber` and `\thepage` at the time of the `\shipout`.

19 A `\linelabel` is allowed only in outer horizontal mode. In outer ver-  
20 tical mode we start a paragraph, and ignore trailing spaces (by fooling  
21 `\@esphack`).

22 (New v4.00) We aim at relaxing the previous condition. We insert a hook  
23 `\@LN@mathhook` and a shorthand `\@LN@postlabel` to support the `mathrefs`  
24 option which allows `\linelabel` in math mode.

25 The next paragraph is no longer valid.

```
26 % The argument of ~\linelabel~ is put into a macro with a
27 % name derived from the number of the allocated float box.
28 % Much of the rest is dummy float setup.
```

29 (/New v4.00)

30 (New v4.11)

```
31 % \def\linelabel#1{%
```

32 I forgot `\linenumbers` today, costed me hours or so.

```
201 \def\@LN@LError{\PackageError{lineno}{%
202 \string\linelabel\space without \string\linenumbers}{%
203 Just see documentation. (New feature v4.11)}\@gobble}
```

1 (New v4.3) Here some things have changed for v4.3. The previous #1  
 2 has been replaced by `\@gobble`. Ensuing, the `\linelabel` error mes-  
 3 sage is re-implemented. I find it difficult to compare efficiency of slight  
 4 alternatives—so choose an easy one. Explicit switching in `\linenumbers`  
 5 and `\nolinenumbers` is an additional command that may better be avoided.

```
204 \newcommand\linelabel{%
205   \ifLineNumbers \expandafter \@LN@linelabel
206   \else          \expandafter \@LN@LError   \fi}
207
208 \gdef\@LN@linelabel#1{%
```

6 `\gdef` for hyperref “symbolically”. (/New v4.11)

```
209 \ifx\protect\@typeset@protect
```

7 ← And a `\linelabel` should never be replicated in a mark or a TOC entry.  
 8 (/New v4.3)

```
210 \ifvmode
211   \ifinner \else
212     \leavevmode \@bsphack \@savsk\p@
213   \fi
214 \else
215   \@bsphack
216 \fi
217 \ifhmode
218   \ifinner
219     \@parmoderr
220   \else
```

9 (New v4.00)

```
221   \@LN@postlabel{#1}%

10 %   \@floatpenalty -\@Mii
11 %   \@next\@currbox\@freelist
12 %   {\global\count\@currbox-2%
13 %   \expandafter\gdef\csname @LNL@the\@currbox\endcsname{#1}}%
14 %   {\@floatpenalty\z@ \@fltovf \def\@currbox{\@tempboxa}}%
15 %   \begingroup
16 %   \setbox\@currbox \color@vbox \vbox \bgroup \end@float
17 %   \endgroup
18 %   \@ignorefalse \@esphack
```

19 (/New v4.00)

```

222     \@esphack

1 (New v4.00) The \@ignorefalse was appropriate before because the
2 \@Esphack in \end@float set \@ignoretrue. Cf. LATEX's \@xympar. (/New
3 v4.00)

223     \fi
224     \else

4 (New v4.00)

225     \@LN@mathhook{#1}%

5 %     \@parmoderr

6 Instead of complaining, you may just do your job. (/New v4.00)

226     \fi
227     \fi
228     }

7 (New v4.00) The shorthand just does what happened with linenox0.sty
8 before ednmath0.sty (New v4.1: now mathrefs option) appeared, and the
9 hook is initialized to serve the same purpose. So errors come just where
10 Stephan had built them in, and this is just the LATEX \marginpar behaviour.

229 \def\@LN@postlabel#1{\g@addto@macro\@LN@labellist{#1\@1t}%
230     \vadjust{\penalty-\@M11bcodepen}}
231 \def\@LN@mathhook#1{\@parmoderr}

11 (/New v4.00)

```

## 5 The appearance of the line numbers

### 13 5.1 Basic code

The line numbers are set as `\tiny\sffamily\arabic{linenumber}`, 10pt left of the text. With options to place it right of the text, or . . .

16 . . . here are the hooks:

```

232 \def\makeLineNumberLeft{%
233   \hss\linenumberfont\LineNumber\hskip\linenumbersep}
234
235 \def\makeLineNumberRight{%
236   \linenumberfont\hskip\linenumbersep\hskip\columnwidth
237   \hb@xt@\linenumberwidth{\hss\LineNumber}\hss}
238
239 \def\linenumberfont{\normalfont\tiny\sffamily}
240
241 \newdimen\linenumbersep
242 \newdimen\linenumberwidth
243
244 \linenumbersep=10pt
245 \linenumberwidth=10pt

```

- 1 Margin switching requires `pagewise` numbering mode, but choosing the left or right margin for the numbers always works.

```

246 \def\switchlinenumbers{\@ifstar
247   {\let\makeLineNumberOdd\makeLineNumberRight
248     \let\makeLineNumberEven\makeLineNumberLeft}%
249   {\let\makeLineNumberOdd\makeLineNumberLeft
250     \let\makeLineNumberEven\makeLineNumberRight}%
251   }
252
253 \def\setmakelinenumbers#1{\@ifstar
254   {\let\makeLineNumberRunning#1%
255     \let\makeLineNumberOdd#1%
256     \let\makeLineNumberEven#1}%
257   {\ifx\c@linenumber\c@runninglinenumber
258     \let\makeLineNumberRunning#1%
259     \else
260       \let\makeLineNumberOdd#1%
261       \let\makeLineNumberEven#1%
262     \fi}%
263   }
264
265 \def\leftlinenumbers{\setmakelinenumbers\makeLineNumberLeft}
266 \def\rightlinenumbers{\setmakelinenumbers\makeLineNumberRight}
267
268 \leftlinenumbers*

```

`\LineNumber` is a hook which is used for the modulo stuff. It is the command  
 4 to use for the line number, when you customize `\makeLineNumber`. Use  
`\thelinenumber` to change the outfit of the digits.

We will implement two modes of operation:

- 7 • numbers running through (parts of) the text

- 1 • `pagewise` numbers starting over with one on top of each page.

Both modes have their own count register, but only one is allocated as a  $\LaTeX$  counter, with the attached facilities serving both.

```
269 \newcounter{linenumber}
270 \newcount\c@pagewiselinenumber
271 \let\c@runninglinenumber\c@linenumber
```

- 4 Only the running mode counter may be reset, or preset, for individual paragraphs. The pagewise counter must give a unique anonymous number for each line.

7 (New v4.3) `\newcounter{linenumber}` was the only `\newcounter` in the whole package, and formerly I was near using `\newcount` instead. Yet `\newcounter` may be quite useful for `\includeonly`. It also supports resetting “subcounters”, but what could these be? Well, `edtable` might introduce a subcounter for columns. (Note that  $\LaTeX$ ’s setting commands would work with `\newcount\c@linenumber` already, apart from this. And perhaps sometimes `\refstepcounter{linenumber}` wouldn’t work—cf. my discussion of `\stepcounter` in subsection 2.4, similarly `\refstep...` would be quite useless. Even the usual redefinitions of `\thelinenumber` would work. It is nice, on the other hand, that `\thelinenumber` is predefined here.  $\LaTeX$ ’s initialization of the value perhaps just serves making clear  $\LaTeX$  counters should always be changed globally.—Shortened and improved the discussion here.)

19 (/New v4.3)

(New v4.22) `\c@linenumber` usually is—globally—incremented by `\stepcounter` (at present), so resetting it locally would raise the save stack problem of  $\TeX$ book p. 301, moreover it would be is useless, there is no hope of keeping the values local (but see subsection 7.2). So I insert `\global:`

22 (/New v4.22)

```
272 \newcommand*\resetlinenumber[1][\@ne]{%
273   \global                               % v4.22
274   \c@runninglinenumber#1\relax}
```

25 (New v4.00)

```
% \newcommand\resetlinenumber[1][1]{\c@runninglinenumber#1}
```

Added `\relax`, being quite sure that this does no harm and is quite important, as with `\setcounter` etc. I consider this a bug fix (although perhaps no user has ever had a problem with this). (/New v4.00)

(v4.22: I had made much fuss about resetting subordinate counters here—  
31 removed, somewhat postponed.)

## 1 5.2 Running line numbers

Running mode is easy, `\LineNumber` and `\theLineNumber` produce `\thelinenumber`, which defaults to `\arabic{linenumber}`, using the `\c@runninglinenumber` counter. This is the default mode of operation.

```

275 \def\makeRunningLineNumber{\makeLineNumberRunning}
276
277 \def\setrunninglinenumbers{%
278   \def\theLineNumber{\thelinenumber}%
279   \let\c@linenumber\c@runninglinenumber
280   \let\makeLineNumber\makeRunningLineNumber
281   }
282
283 \setrunninglinenumbers\resetlinenumber

```

## 5.3 Pagewise line numbers

Difficult, if you think about it. The number has to be printed when there is no means to know on which page it will end up, except through the aux-file. My solution is really expensive, but quite robust.

With version v2.00 the hashsize requirements are reduced, because we do not need one controlsequence for each line any more. But this costs some computation time to find out on which page we are.

`\makeLineNumber` gets a hook to log the line and page number to the aux-file. Another hook tries to find out what the page offset is, and subtracts it from the counter `\c@linenumber`. Additionally, the switch `\ifoddNumberedPage` is set true for odd numbered pages, false otherwise.

```

284 \def\setpagewiselinenumbers{%
285   \let\theLineNumber\thePagewiseLineNumber
286   \let\c@linenumber\c@pagewiselinenumber
287   \let\makeLineNumber\makePagewiseLineNumber
288   }
289
290 \def\makePagewiseLineNumber{\logtheLineNumber\getLineNumber
291   \ifoddNumberedPage
292     \makeLineNumberOdd
293   \else
294     \makeLineNumberEven
295   \fi
296   }

```

16 Each numbered line gives a line to the aux file

```
\@LN{<line>}{<page>}
```

1 very similar to the `\newlabel` business, except that we need an arabic representation of the page number, not what there might else be in `\thepage`.

```
297 \def\logtheLineNumber{\protected@write\auxout-}{%
```

(New v4.00) (UL) As Daniel Doherty observed, the earlier line

```
4 % \string\@LN{\the\c@linenumber}\noexpand\the\c@page}}
```

here may lead into an infinite loop when the user resets the page number (think of `\pagenumbering`, e.g.). Stephan and I briefly discussed  
7 the matter and decided to introduce a “physical”-page counter to which `\logtheLineNumber` refers. It was Stephan’s idea to use `\c@page` for reliably augmenting the “physical”-page counter. However, this relies on the  
10 output routine once doing `\stepcounter{page}`. Before Stephan’s suggestion, I had thought of appending the stepping to L<sup>A</sup>T<sub>E</sub>X’s `\@outputpage`.—So the macro definition ends as follows.

```
298 \string\@LN{\the\c@linenumber}{%
```

13 (New v4.2) The ‘truepage’ counter must start with `\c@` so it works with `\include`, and the `\@addtoreset` below is needed for the same purpose.

```
299 \noexpand\the\c@LN@truepage}}}
```

```
300
```

```
301 \newcount\c@LN@truepage
```

```
302 \g@addto@macro\c@page{\global\advance\c@LN@truepage\@ne}
```

```
303 \@addtoreset{LN@truepage}{\@ckpt}
```

(/New v4.2) I had thought of offering more features of a L<sup>A</sup>T<sub>E</sub>X counter.  
16 However, the user should better *not* have access to this counter. `\c@page` should suffice as a pagewise master counter.—To be sure, along the present lines the user *can* manipulate `\c@LN@truepage` by `\stepcounter{page}`.  
19 E.g., she might do this in order to manually insert a photograph. Well, seems not to harm.

The above usage of `\g@addto@macro` and `\c@page` may be not as stable as Stephan intended. His proposal used `\xdef` directly. But he used `\c@page` as well, and who knows ... And as to `\g@addto@macro`, I have introduced it for list macros anyway. (/UL) (/New v4.00)

25 From the aux-file we get one macro `\LN@P{page}` for each page with line numbers on it. This macro calls four other macros with one argument each. These macros are dynamically defined to do tests and actions, to find out on  
28 which page the current line number is located.

We need sort of a pointer to the first page with line numbers, initialized to point to nothing:

```

304 \def\LastNumberedPage{first}
305 \def\LN@Pfirst{\nextLN\relax}

```

1 The four dynamic macros are initialized to reproduce themselves in an `\xdef`

```

306 \let\lastLN\relax % compare to last line on this page
307 \let\firstLN\relax % compare to first line on this page
308 \let\pageLN\relax % get the page number, compute the line number
309 \let\nextLN\relax % move to the next page

```

During the end-document run through the aux-files, we disable `\@LN`. I may put in a check here later, to give a rerun recommendation.

```

310 \AtEndDocument{\let\@LN@gobbletwo}

```

4 Now, this is the tricky part. First of all, the whole definition of `\@LN` is grouped, to avoid accumulation on the save stack. Somehow `\csname<cs>\endcsname` pushes an entry, which stays after an `\xdef` to that

7 `<cs>`.

If `\LN@P<page>` is undefined, initialize it with the current page and line number, with the *pointer-to-the-next-page* pointing to nothing. And the

10 macro for the previous page will be redefined to point to the current one.

If the macro for the current page already exists, just redefine the *last-line-number* entry.

13 Finally, save the current page number, to get the pointer to the following page later.

```

311 \def\@LN#1#2{\expandafter\@LN
312             \csname LN@P#2\@LN@column\expandafter\endcsname
313             \csname LN@P#2\endcsname
314             {#1}{#2}}
315
316 \def\@LN#1#2#3#4{\ifx#1\relax
317   \ifx#2\relax\gdef#2{#3}\fi
318   \expandafter\@LN\csname LN@P\LastNumberedPage\endcsname#1%
319   \xdef#1{\lastLN{#3}\firstLN{#3}%
320           \pageLN{#4}{\@LN@column}{#2}\nextLN\relax}%
321 \else
322   \def\lastLN##1{\noexpand\lastLN{#3}}%
323   \xdef#1{#1}%
324 \fi
325 \xdef\LastNumberedPage{#4C\@LN@column}}

```

The previous page macro gets its pointer to the current one, replacing the

16 `\relax` with the cs-token `\LN@P<page>`.

```

326 \def\@@LN#1#2{{\def\nextLN##1{\noexpand\nextLN\noexpand#2}%
327         \xdef#1{#1}}}

```

- 1 Now, to print a line number, we need to find the page, where it resides. This  
 will most probably be the page where the last one came from, or maybe the  
 next page. However, it can be a completely different one. We maintain a  
 4 cache, which is `\let` to the last page's macro. But for now it is initialized  
 to expand `\LN@first`, where the pointer to the first numbered page has been  
 stored in.

```

328 \def\NumberedPageCache{\LN@Pfirst}

```

- 7 To find out on which page the current `\c@linenumber` is, we define the four  
 dynamic macros to do something usefull and execute the current cache macro.  
`\lastLN` is run first, testing if the line number in question may be on a later  
 10 page. If so, disable `\firstLN`, and go on to the next page via `\nextLN`.

```

329 \def\testLastNumberedPage#1{\ifnum#1<\c@linenumber
330     \let\firstLN@gobble
331     \fi}

```

- Else, if `\firstLN` finds out that we need an earlier page, we start over  
 from the beginning. Else, `\nextLN` will be disabled, and `\pageLN` will run  
 13 `\gotNumberedPage` with four arguments: the first line number on this col-  
 umn, the page number, the column number, and the first line on the page.

```

332 \def\testFirstNumberedPage#1{\ifnum#1>\c@linenumber
333     \def\nextLN##1{\testNextNumberedPage\LN@Pfirst}%
334     \else
335         \let\nextLN@gobble
336         \def\pageLN{\gotNumberedPage{#1}}%
337     \fi}

```

- We start with `\pageLN` disabled and `\nextLN` defined to continue the search  
 16 with the next page.

```

338 \long\def \@gobblethree #1#2#3{}
339
340 \def\testNumberedPage{%
341     \let\lastLN\testLastNumberedPage
342     \let\firstLN\testFirstNumberedPage
343     \let\pageLN@gobblethree
344     \let\nextLN\testNextNumberedPage
345     \NumberedPageCache
346 }

```

1 When we switch to another page, we first have to make sure that it is there.  
 If we are done with the last page, we probably need to run T<sub>E</sub>X again, but for  
 the rest of this run, the cache macro will just return four zeros. This saves a  
 4 lot of time, for example if you have half of an aux-file from an aborted run, in  
 the next run the whole page-list would be searched in vain again and again  
 for the second half of the document.

7 If there is another page, we iterate the search.

```

347 \def\testNextNumberedPage#1{\ifx#1\relax
348   \global\def\NumberedPageCache{\gotNumberedPage0000}%
349   \PackageWarningNoLine{lineno}%
350   {Line number reference failed, re-run to get it right}%
351 \else
352   \global\let\NumberedPageCache#1%
353 \fi
354 \testNumberedPage
355 }
```

To separate the official hooks from the internals there is this equivalence, to  
 hook in later for whatever purpose:

Let's see if  
 it finds the  
 label on  
 page 22,  
 line 4, and  
 back here  
 on page  
 31, line 8.

```

356 \let\getLineNumber\testNumberedPage
```

10 So, now we got the page where the number is on. We establish if we are on  
 an odd or even page, and calculate the final line number to be printed.

```

357 \newif\ifoddNumberedPage
358 \newif\ifcolumnwiselinenumbers
359 \columnwiselinenumbersfalse
360
361 \def\gotNumberedPage#1#2#3#4{\oddNumberedPagefalse
362 \ifodd \if@twocolumn #3\else #2\fi\relax\oddNumberedPagetrue\fi
363 \advance\c@linenumber\@ne
364 \ifcolumnwiselinenumbers
365   \subtractlinenumberoffset{#1}%
366 \else
367   \subtractlinenumberoffset{#4}%
368 \fi
369 }
```

13 You might want to run the pagewise mode with running line numbers, or  
 you might not. It's your choice:

```

370 \def\runningpagewiselinenumbers{%
371   \let\subtractlinenumberoffset\@gobble
372 }
373
```

```

374 \def\realpagewiselinenumbers{%
375   \def\subtractlinenumberoffset##1{\advance\c@linenumber-##1\relax}%
376   }
377
378 \realpagewiselinenumbers

```

- 1 For line number references, we need a protected call to the whole procedure, with the requested line number stored in the `\c@linenumber` counter. This is what gets printed to the aux-file to make a label:

```

379 \def\thePagewiseLineNumber{\protect
380   \getpagewiselinenumber{\the\c@linenumber}}%

```

- 4 And here is what happens when the label is referred to:

```

381 \def\getpagewiselinenumber#1{%
382   \c@linenumber #1\relax\testNumberedPage
383   \thelinenumber
384   }}

```

A summary of all per line expenses:

- 7 **CPU:** The `\output` routine is called for each line, and the page-search is done.

**DISK:** One line of output to the aux-file for each numbered line

- 10 **MEM:** One macro per page. Great improvement over v1.02, which had one control sequence per line in addition. It blew the hash table after some five thousand lines.

## 5.4 Twocolumn mode (New v3.06)

- 13 Twocolumn mode requires another patch to the `\output` routine, in order to print a column tag to the .aux file.

```

385 \AtBeginDocument{% v4.2, revtex4.cls (e.g.).
386   % <- TODO v4.4+: Or better in \LineNoLaTeXOutput!?
387   \let\@LN@orig@makecol\@makecol
388   }
389
390 \def\@LN@makecol{%
391   \@LN@orig@makecol
392   \setbox\@outputbox \vbox{%
393     \boxmaxdepth \@maxdepth
394     \protected@write\@auxout}{%

```

```

395     \string\def\string\@LN@column{\if@firstcolumn1\else2\fi}%
396     }%
397     \box\@outputbox
398     }% \vbox
399 } %% TODO cf. revtexln.sty.
400
401 \def\@LN@column{1}
402
403 \ifdefined\IfFormatAtLeastTF
404   \IfFormatAtLeastTF{2025-06-01}{
405     % disable \@LN@makecol
406     \AtBeginDocument{%
407       \let\@LN@makecol\@makecol
408     }
409     % replace with hook code
410     \AddToHook{build/column/after}{%
411       \setbox\@outputbox \vbox to\ht\@outputbox{%
412         \boxmaxdepth \@maxdepth
413         \protected@write\@auxout{}{%
414           \string\def\string\@LN@column{\if@firstcolumn1\else2\fi}%
415         }%
416         \unvbox\@outputbox
417       }%
418     }
419   }{
420   }
421 \fi

```

## 1 5.5 Numbering modulo $m$ , starting at $f$

Most users want to have only one in five lines numbered. `\LineNumber` is supposed to produce the outfit of the line number attached to the line, while `\thelinenumber` is used also for references, which should appear even if they are not multiples of five.

(New v4.00) Moreover, some users want to control which line number should be printed first. Support of this is now introduced here—see `\firstlinenumber` below.—`numline.sty` by Michael Jaegermann and James Fortune offers controlling which *final* line numbers should not be printed. What is it good for? We ignore this here until some user demands it.—Peter Wilson’s `ledmac.sty` offers much different choices of line numbers to be printed, due to Wayne Sullivan. (/New v4.00)

(New v4.22) `\c@linenumbermodulo` is rendered a fake counter, as discussed since v4.00. So it can no longer be set by `\setcounter`. `\modulolinenumbers` serves this purpose. Well, does anybody want to do what worked with `\addtocounter`? (Then please tell me.)—At least, `\value`

1 still works. For the same purpose I rename the fake ‘firstlinenumber’ counter  
 \n@... to \c@.... (/New v4.22)

```
% \newcount\c@linenumbermodulo % removed for v4.22
```

4 (New v4.00)  
 \themodulolinenumber waits for being declared \LineNumber by  
 \modulolinenumbers. (This has been so before, no change.) Here is how it  
 7 looked before:

```
% \def\themodulolinenumber{{\@tempcnta\c@linenumber
% \divide\@tempcnta\c@linenumbermodulo
10 % \multiply\@tempcnta\c@linenumbermodulo
% \ifnum\@tempcnta=\c@linenumber\thelinenumber\fi
% }}
```

13 (UL) This was somewhat slow. This arithmetic happens at every line. This  
 time I tend to declare an extra line counter (as opposed to my usual rec-  
 ommendations to use counters as rarely as possible) which is stepped every  
 16 line. It could be incremented in the same way as \c@LN@truepage is incre-  
 mented via \c1@page! This is another point in favour of {linenumber} being  
 a L<sup>A</sup>T<sub>E</sub>X counter! When this new counter equals \c@linenumbermodulo, it is  
 19 reset, and \thelinenumber is executed.—It gets much slower by my support  
 of controlling the first line number below. I should improve this.—On the  
 other hand, time expense means very little nowadays, while the number of  
 22 T<sub>E</sub>X counters still is limited.

For the same purpose, moreover, attaching the line number box could be  
 intercepted earlier (in \MakeLineNo), without changing \LineNumber. How-  
 25 ever, this may be bad for the latter’s announcement as a wizard interface in  
 section 10. (/UL)

Here is the new code. It is very near to my lnoptch.sty code which  
 28 introduced the first line number feature before.—I add starting with a \relax  
 which is so often recommended—without understanding this really. At least,  
 it will not harm.—Former group braces appear as \begingroup/\endgroup  
 31 here.

```
422 \def\themodulolinenumber{\relax
423 \ifnum\c@linenumber<\c@firstlinenumber \else
424 \begingroup
425 \@tempcnta\c@linenumber
426 \advance\@tempcnta-\c@firstlinenumber
427 \divide\@tempcnta\c@linenumbermodulo
428 \multiply\@tempcnta\c@linenumbermodulo
429 \advance\@tempcnta\c@firstlinenumber
```

```

430     \ifnum\@tempcnta=\c@linenumber \thelinenumber \fi
431   \endgroup
432   \fi
433 }

```

1 (/New v4.00)

The user command to set the modulo counter: (New v4.31) ... a star variant is introduced to implement Hillel Chayim Yisraeli's idea to print the first line number after an interruption of the edited text by some editor's text, regardless of the modulo. If it is 1, it is printed only with `\firstlinenumber{1}`. I.e., you use `\modulolinenumbers*` for the new feature, without the star you get the simpler behaviour that we have had so far. And you can switch back from the refined behaviour to the simple one by using `\modulolinenumbers` without the star.—This enhancement is accompanied by a new package option `modulo*` which just executes `\modulolinenumbers*` (subsection 6.4).—'With `\firstlinenumber{1}`' exactly means: '1' is printed if and only if the last `\firstlinenumber` before or in the paragraph that follows the "interruption" has argument '1' (or something *expanding* to '1', or (to) something that T<sub>E</sub>X "reads" as 1, e.g.: a T<sub>E</sub>X count register storing 1).—At present, this behaviour may be unsatisfactory with pagewise line-numbering ... I'll make an experimental extra package if someone complains ...

```

434 \newcommand\modulolinenumbers{%
435   \@ifstar
436     {\def\@LN@maybe@moduloresume{%
437       \global\let\@LN@maybe@normalLineNumber
438         \@LN@normalLineNumber}%
439       \@LN@modulolinenos}%
440     {\let\@LN@maybe@moduloresume\relax \@LN@modulolinenos}%
441 }
442
443 \global\let\@LN@maybe@normalLineNumber\relax
444 \let\@LN@maybe@moduloresume\relax
445 \gdef\@LN@normalLineNumber{%
446   \ifnum\c@linenumber=\c@firstlinenumber \else
447     \ifnum\c@linenumber>\@ne
448       \def\LineNumber{\thelinenumber}%
449     \fi
450   \fi

```

`\def` instead of `\let` enables taking account of a redefinition of `\thelinenumber` in a present numbering environment (e.g.).

```

451   \global\let\@LN@maybe@normalLineNumber\relax}

```

- 1 Instead of changing `\LineNumber` directly by `LN@moduloresume`, these tricks  
 enable `\modulolinenumbers*` to act as locally as I can make it. I don't know  
 how to avoid that the output routine switches back to the normal modulo  
 4 behaviour by a global change. (An `\aftergroup` may fail in admittedly  
 improbable cases.)

```
452 \newcommand*\LN@modulolinenos[1][\z@]{%
```

The definition of this macro is that of the former `\modulolinenumbers`.  
 7 (/New v4.31)

```
453 \let\LineNumber\thomodulolinenumner
454 \ifnum#1>\@ne
455   \chardef                % v4.22, note below
456   \c@linenumbermodulo#1\relax
457 \else\ifnum#1=\@ne
```

```
% \def\LineNumber{\thelinenumner}%
```

- (New v4.00) I am putting something here to enable `\firstlinenumner` with  
 10 `\c@linenumbermodulo = 1`. With `lnopatch.sty`, a trick was offered for this  
 purpose. It is now obsolete.

```
458 \def\LineNumber{\LN@ifgreat\thelinenumner}%
```

(/New v4.00)

```
459 \fi\fi
460 }
```

- 13 (New v4.00) The default of `\LN@ifgreat` is

```
461 \let\LN@ifgreat\relax
```

The previous changes as soon as `\firstlinenumner` is used:

```
462 \newcommand*\firstlinenumner[1]{%
463 \chardef\c@firstlinenumner#1\relax
```

- No counter, little values allowed only—OK?—(UL) The change is local—  
 16 OK? The good thing is that `\global\firstlinenumner{<number>}` works.  
 Moreover, `\modulolinenumbers` acts locally as well. (/UL)  
 (New v4.31)

```

464 \let\@LN@ifgreat\@LN@ifgreat@critical}
465
466 \def\@LN@ifgreat@critical{%
467   \ifnum\c@linenumber<\c@firstlinenumber
468     \expandafter \@gobble
469   \fi}%

```

1 (/New v4.31)

The default value of `\c@firstlinenumber` is 0. This is best for what one would expect from modulo printing.

```

470 \let\c@firstlinenumber=\z@

```

4 For usage and effects of `\modulolinenumbers` and `\firstlinenumbers`, please consult section 10. Two details on `\firstlinenumbers` here: (i) `\firstlinenumber` acts on a paragraph if and only if (a) the paragraph  
7 is broken into lines “in line-numbering mode” (after `\linenumbers`, e.g.); (b) it is the last occurrence of a `\firstlinenumbers` before or in the paragraph. (The practical applications of this that I can imagine don’t seem  
10 appealing to me.) Cf. the explanation above of how `\modulolinenumbers` and `\firstlinenumbers` interact—for this and for (ii), which is concerned with possible arguments for `\firstlinenumbers`.

13 Note that the line numbers of the present section demonstrate the two devices. (/New v4.00)

```

471 \chardef\c@linenumbermodulo=5      % v4.2; ugly?
472 \modulolinenumbers[1]

```

(New v4.22) The new implementation through `\chardef` decreases the func-  
16 tionality and raises certain compatibility problems. I face this without fear. The maximum modulo value is now 255. I expect that this suffices for usual applications. However, some users have “abused” `lineno.sty` to get  
19 `ednotes.sty` features without line numbers, so have set the modulo to a value beyond the total number of lines in their edition. This ought to be replaced by `\let\makeLineNumber\relax`. (/New v4.22)

## 22 6 Package options

(New v4.1) The last heading formerly was the heading of what is now sub-  
25 section 6.4. The options declared there were said to execute user commands only. This was wrong already concerning `displaymath` and `hyperref`. At least, however, these options were no or almost no occasion to skip definitions or allocations. This is different with the options that we now insert.

## 1 6.1 Extended referencing to line numbers. (v4.2)

This subsection explains and declares package option `addpageno`.

If a line to whose number you refer by `\ref` is not on the present page,  
 4 it may be useful to add the number of the page on which the line occurs—  
 and perhaps it should not be added otherwise. In general, you could use  
 the Standard L<sup>A</sup>T<sub>E</sub>X package `varioref` for this. However, the latter usually  
 7 produces verbose output like ‘on the preceding page’— unless customized—,  
 while in critical editions, e.g., one may prefer just adding the page number  
 and some mark on the left of the line number, irrespectively of how far the  
 10 page is apart etc. To support this, package option `addpageno` provides a  
 command `\vpagelinerref` to be used in place of `\ref`. This produces, e.g.,  
 ‘34.15’ when referring to line 15 on page 34 while the present page is not  
 13 34. You can customize the outcome, see the package file `vplref.sty` where  
 the code and further details are. You may conceive of `\vpagelinerref` as a  
 certain customization of `varioref`’s `\vref`.

16 This implies that option `addpageno` requires the files `vplref.sty` and  
`varioref.sty`. `addpageno` automatically loads both of them. Yet you can  
 also load `varioref.sty` on your own to use its package options.

19 Of course, you might better introduce a shorter command name for  
`\vpagelinerref` for your work, while we cannot predict here what shorthand  
 will fit your work. E.g., `\newcommand{\lref}{\vpagelinerref}`.

22 If you really want to add the page number in *any* case, use, e.g., some  
`\myref` instead of `\ref`, after

```
newcommand*\myref{\pageref{#1}.\ref{#1}}
```

or what you like. You don’t need the `addpageno` option in this case.

25 `addpageno` is due to a suggestion by Sergei Mariev.

```
473 \DeclareVoidOption{addpageno}{%
474 \AtEndOfPackage{\RequirePackage{vplref}[2005/04/25]}}
```

## 6.2 `\linelabel` in math mode

We have made some first steps towards allowing `\linelabel` in math mode.

28 Because our code for this is presently experimental, we leave it to the user to  
 decide for the experiment by calling option `mathrefs`. We are in a hurry now  
 and thus leave the code, explanations, and discussion in the separate package  
 31 `ednmath0.sty`. Maybe we later find the time to improve the code and move  
 the relevant content of `ednmath0.sty` to here. The optimal situation would  
 be to define `\linelabel` from the start so it works in math mode, omitting  
 34 the `mathrefs` option.

1     Actually, this package even provides adjustments for analogously allowing `ednotes.sty` commands in math mode. Loading the package is postponed to `\AtBeginDocument` when we know whether these adjustments are needed.

```
475 \DeclareVoidOption{mathrefs}{\AtBeginDocument
476   {\RequirePackage{edmath0}[2004/08/20]}}
```

### 4 6.3 Arrays, tabular environments (Revised v4.11)

This subsection explains and declares package options `edtable`, `longtable`, and `nolongtablepatch`.

7     The standard  $\LaTeX$  tabular environments come as single boxes, so the `lineno.sty` versions before v4.00 treated them as (parts of) single lines, printing (at most) one line number beside each and stepping the line number counter once only. Moreover, `\linelabels` got lost. Of course, tables are usually so high that you will want to treat each row like a line. (Christian Tapp even desires that the lines of table entries belonging to a single row are treated like ordinary lines.) Footnotes get lost in such environments as well, which was bad for `ednotes.sty`.

We provide adjustments to count lines, print their numbers etc. as desired at least for *some*  $\LaTeX$  tabular environments. (Like with other details, “some” is to some extent explained in `edtable.sty`.) We do this similarly as with option `mathrefs` before. We leave code and explanations in the separate package `edtable.sty`. (For wizards: this package provides adjustments for `ednotes.sty` as well. However, in the present case we don’t try to avoid them unless `ednotes.sty` is loaded.) Package option `edtable` defines—by loading `edtable.sty`—an environment `{edtable}` which is able to change some  $\LaTeX$  tabular environments with the desired effects. (v4.11: `edtable.sty` v1.3 counts  $\LaTeX$ ’s `{array}` [etc.] as a “tabular environment” as well.)

25     The `{edtable}` environment doesn’t help with `longtable.sty`, however. To make up for this, `{longtable}` is adjusted in a different way—and this happens only when another `lineno.sty` option `longtable` is called. In this case, option `edtable` needn’t be called explicitly: option `longtable` works as if `edtable` had been called.

Now, we are convinced that vertical spacing around `{longtable}` works wrongly—see  $\LaTeX$  bugs database tools/3180 and 3485, or see explanations in the package `ltabptch.sty` (which is to be obtained from CTAN folder `/macros/latex/ltabptch`). Our conviction is so strong that the `longtable` option loads—after `longtable.sty`—the patch package `ltabptch.sty`. If the user doesn’t want this (maybe preferring her own arrangement with the vertical spacing), she can forbid it by calling `nolongtablepatch`.

1 The following code just collects some choices, which are then executed  
 in section 6.7. We use an `\if...` without `\newif` since `\if...true` and  
`\if...false` would occur at most two times and only within the present  
 4 package. (`\AtEndOfClass{\RequirePackage{edtable}}` could be used in-  
 stead, I just overlooked this. Now I don't change it because it allows to  
 change the version requirement at one place only.)

```
477 \let\if@LN@edtable\iffalse
478
479 \DeclareVoidOption{edtable}{\let\if@LN@edtable\iftrue}
480
481 \DeclareVoidOption{longtable}{\let\if@LN@edtable\iftrue
482   \PassOptionsToPackage{longtable}{edtable}}
483
484 \DeclareVoidOption{nolongtablepatch}{%
485   \PassOptionsToPackage{nolongtablepatch}{edtable}}

7 (/New v4.1)
```

## 6.4 Switch among settings

There is a bunch of package options that execute user commands only.

10 Options `left` (`right`) put the line numbers on the left (right) margin.  
 This works in all modes. `left` is the default.

```
486 \DeclareVoidOption{left}{\leftlinenumbers*}
487
488 \DeclareVoidOption{right}{\rightlinenumbers*}
```

13 Option `switch` (`switch*`) puts the line numbers on the outer (inner) margin  
 of the text. This requires running the pagewise mode, but we turn off the  
 page offset subtraction, getting sort of running numbers again. The `pagewise`  
 option may restore true pagewise mode later.

```
489 \DeclareVoidOption{switch}{\setpagewiselinenumbers
490   \switchlinenumbers
491   \runningpagewiselinenumbers}
492
493 \DeclareVoidOption{switch*}{\setpagewiselinenumbers
494   \switchlinenumbers*%
495   \runningpagewiselinenumbers}
```

16 In twocolumn mode, we can switch the line numbers to the outer margin,  
 and/or start with number 1 in each column. Margin switching is covered by  
 the `switch` options.

```

496 \DeclareVoidOption{columnwise}{\setpagewiselinenumbers
497                               \columnwiselinenumberstrue
498                               \realpagewiselinenumbers}

```

- 1 The options `pagewise` and `running` select the major line number mechanism. `running` line numbers refer to a real counter value, which can be reset for any paragraph, even getting multiple paragraphs on one page starting with
- 4 line number one. `pagewise` line numbers get a unique hidden number within the document, but with the opportunity to establish the page on which they finally come to rest. This allows the subtraction of the page offset, getting
- 7 the numbers starting with 1 on top of each page, and margin switching in twoside formats becomes possible. The default mode is `running`.

The order of declaration of the options is important here `pagewise` must

10 come after `switch`, to override running pagewise mode. `running` comes last, to reset the running line number mode, e.g, after selecting margin switch mode for `pagewise` running. Once more, if you specify all three of the options

13 [`switch,pagewise,running`], the result is almost nothing, but if you later say `\pagewiselinenumbers`, you get margin switching, with real pagewise line numbers.

```

499 \DeclareVoidOption{pagewise}{\setpagewiselinenumbers
500                               \realpagewiselinenumbers}
501
502 \DeclareVoidOption{running}{\setrunninglinenumbers}

```

- 16 The option `modulo` causes only those linenumbers to be printed which are multiples of five.

```

503 \DeclareVoidOption{modulo}{\modulolinenumbers\relax}

```

Option `modulo*` modifies `modulo` in working like `\modulolinenumbers*`—see

19 section 10.

```

504 \DeclareVoidOption{modulo*}{\modulolinenumbers*\relax}

```

The package option `mathlines` switches the behavior of the `{linenomath}` environment with its star-form. Without this option, the `{linenomath}`

22 environment does not number the lines of the display, while the star-form does. With this option, its just the opposite.

```

505 \DeclareVoidOption{mathlines}{\linenumberdisplaymath}

```

1 `displaymath` now calls for wrappers of the standard L<sup>A</sup>T<sub>E</sub>X display math environment. This was previously done by `mlineno.sty`.

(New v4.3) Option ‘displaymath’ becomes default according to Erik  
 4 Luijten’s suggestion. I was finally convinced of this as soon as I discovered how to avoid a spurious line number above `\begin{linenomath}` (subsection 3.3). `\endlinenomath` provides `\ignorespaces`, so what could go  
 7 wrong now?

```
506 \DeclareVoidOption{displaymath}{%
507   \PackageWarningNoLine{lineno}{Option [displaymath] is obsolete}}
```

(/New v4.3)

(New v5.3) Options ‘sep’ and ‘width’ set `\linenumbersep` (the separation  
 10 of the line number to the text) and `\linenumberwidth` (the width of the line number box on the right margin) respectively; see section 10.1.

```
508 \DeclareStringOption[\linenumbersep]{sep}
509 \DeclareStringOption[\linenumberwidth]{width}
510 \AtBeginDocument{%
511   \linenumbersep=\lineno@sep%
512   \linenumberwidth=\lineno@width%
513 }
```

(/New v5.3)

## 13 6.5 Compatibility with hyperref

The `hyperref` package, via `nameref`, requires three more groups in the second argument of a `\newlabel`. Well, why shouldn’t it get them? (New  
 16 v3.07) The presence of the `nameref` package is now detected automatically `\AtBeginDocument`. (/New v3.07) (Fixed in v3.09) We try to be smart, and test `\AtBeginDocument` if the `nameref` package is loaded, but `hyperref`  
 19 postpones the loading of `nameref` too, so this is all in vain.

(New v4.3) But we can also test at the first `\linelabel`. Regarding the error-message for misplaced `\linelabel` from v4.11: previously,  
 22 `\linenumbers` rendered `\linelabel` the genuine version of `\linelabel` from the start on. This doesn’t work now, since `\@LN@linelabel` may change its meaning after the first `\linenumbers` and before a next one (if there is some).  
 25 (/New v4.3)

```
514 \DeclareVoidOption{hyperref}{%
515   \PackageWarningNoLine{lineno}{Option [hyperref] is obsolete}}
516
517 \AtBeginDocument{%
518   \@ifpackageloaded{nameref}{%
```



## 1 6.7 Execute options

We stop declaring options and execute the ones that are called by the user.  
(/New v4.1)

```
534 \ProcessKeyvalOptions*
```

4 (New v4.1) Now we know whether `edtable.sty` is wanted and (if it is) with which options it is to be called.

```
535 \if@LN@edtable \RequirePackage{edtable}[2005/03/07] \fi
```

(/New v4.1)

## 7 Package extensions

Some of the extensions in this section were previously supplied in separate `.sty` files.

### 10 7.1 displaymath

(New v4.3) From now on, you no longer need to type the `{linenomath}` environment with the `\[`, `{equation}`, and `{eqnarray}` environments—and you no longer need to use the former package option `displaymath` for this feature. (/New v4.3)

The standard L<sup>A</sup>T<sub>E</sub>X display math environments are wrapped in a `{linenomath}` environment.

(New 3.05) The `[fleqn]` option of the standard L<sup>A</sup>T<sub>E</sub>X classes defines the display math environments such that line numbers appear just fine. Thus, we need not do any tricks when `[fleqn]` is loaded, as indicated by presents of the `\mathindent` register. (/New 3.05)

(New 3.05a) for `{eqnarray}`s we rather keep the old trick. (/New 3.05a)

(New 3.08) Wrap `\[` and `\]` into `{linenomath}`, instead of `{displaymath}`. Also save the definition of `\equation`, instead of replicating the current L<sup>A</sup>T<sub>E</sub>X definition. (/New 3.08)

```
536 \@ifundefined{mathindent}{
537
538 \let\LN@displaymath\[%
539 \let\LN@enddisplaymath\]%
540 \renewcommand\[{ \begin{linenomath}\LN@displaymath}%
541 \renewcommand\]{ \LN@enddisplaymath\end{linenomath}}%
```

```

542 \let\LN@equation\equation
543 \let\LN@endequation\endequation
544 \renewenvironment{equation}%
545     {\linenomath\LN@equation}%
546     {\LN@endequation\endlinenomath}%
547
548 }{}% \ifundefined{mathindent} -- 3rd arg v4.2, was \par!
549
550 \let\LN@eqnarray\eqnarray
551 \let\LN@endeqnarray\endeqnarray
552 \renewenvironment{eqnarray}%
553     {\linenomath\LN@eqnarray}%
554     {\LN@endeqnarray\endlinenomath}%

```

- 1 (UL) Indeed. The L<sup>A</sup>T<sub>E</sub>X macros are saved for unnumbered mode, which is detected by `\linenomath`. (/UL)

## 7.2 Line numbers in internal vertical mode

- 4 The command `\internallinenumbers` adds line numbers in internal vertical mode, but with limitations: we assume fixed baseline skip.

(v4.22) v3.10 provided a global (`\global\advance`) as well as a local version (star-form, using `\c@internallinenumbers`). `\resetlinenumbers` acted locally and was here used with the global version—save stack danger, T<sub>E</sub>Xbook p. 301—in v4.00 I disabled the global version therefore.

- 10 Now I find that it is better to keep a global version, and the now global `\resetlinenumbers` is perfect for this. The global version allows continuing the “internal” numbers in the ensuing “external” text, and—unless reset by brackets argument—continuing the above series of line numbers. As
- 13 with v3.10, the local version always starts with line number one. A new `\@LN@iglobal` steps `\globally` in the global version, otherwise it is `\relax`.
- 16 (I also remove all my stupid discussions as of v4.00. And I use `\newcommand`.)

(v4.22)

```

555 \let\@LN@iglobal\global % v4.22
556
557 \newcommand\internallinenumbers{\setrunninglinenumbers
558     \let\@par\internallinenumberspar
559     \ifx\@par\@@par\let\@par\internallinenumberspar\fi
560     \ifx\par\@@par\let\par\internallinenumberspar\fi
561     \ifx\@par\linenumberspar\let\@par\internallinenumberspar\fi
562     \ifx\par\linenumberspar\let\par\internallinenumberspar\fi
563     \@ifnextchar[{\resetlinenumbers}%]
564         {\@ifstar{\let\c@linenumbers\c@internallinenumbers
565         \let\@LN@iglobal\relax % v4.22

```

```

566         \c@linenumber\@ne}{}}%
567     }
568
569 \let\endinternallinenumbers\endlinenumbers
570 \@namedef{internallinenumbers*}{\internallinenumbers*}
571 \expandafter\let\csname endinternallinenumbers*\endcsname\endlinenumbers
572
573 \newcount\c@internallinenumbers
574 \newcount\c@internallinenumbers
575
576 \newcommand\internallinenumberspar{%
577     \ifvmode\@@@par\else\ifinner\@@@par\else\@@@par
578     \begingroup
579         \c@internallinenumbers\prevgraf
580         \setbox\@tempboxa\hbox{\vbox{\makeinternalLinenumbers}}}%
581         \ht\@tempboxa\z@
582         \ifdim\prevdepth=-1000pt
583             % \nointerlineskip is already set so we don't need set it again
584             % (and we shouldn't back up)
585         \else
586             \dp\@tempboxa\prevdepth
587             \nobreak\vskip-\prevdepth
588             \nointerlineskip
589         \fi
590     \box\@tempboxa
591 \endgroup
592 \fi\fi
593 }
594
595 \newcommand\makeinternalLinenumbers{%
596     \ifnum\c@internallinenumbers>\z@           % v4.2
597     \hb@xt@\z@{\makeLineNumber}%
598     \@LN@iglobal                               % v4.22
599     \advance\c@linenumber\@ne
600     \advance\c@internallinenumbers\m@ne
601     \expandafter\makeinternalLinenumbers\fi
602 }
603 % TODO v4.4+: star: line numbers right!? cf. lncapt.sty

```

### 1 7.3 Line number references with offset

This extension defines macros to refer to line numbers with an offset, e.g., to refer to a line which cannot be labeled directly (display math). This was formerly known as `rlineno.sty`.

To refer to a pagewise line number with offset:

$$\backslash\text{lineref}p[\langle\text{OFFSET}\rangle]{\langle\text{LABEL}\rangle}$$

1 To refer to a running line number with offset:

```
\linerefr[⟨OFFSET⟩]{⟨LABEL⟩}
```

To refer to a line number labeled in the same mode as currently selected:

```
4 \lineref[⟨OFFSET⟩]{⟨LABEL⟩}
```

```
604 \newcommand\lineref{%
605   \ifx\c@linenumber\c@runninglinenumber
606     \expandafter\linerefr
607   \else
608     \expandafter\linerefp
609   \fi
610 }
611
612 \newcommand*\linerefp[2][\z@]{%
613   \let\@thelinenumber\thelinenumber
614   \edef\thelinenumber{\advance\c@linenumber#1\relax
615                       \noexpand\@thelinenumber}%
616   \ref{#2}%
617 }}
```

This goes deep into L<sup>A</sup>T<sub>E</sub>X's internals.

```
618 \newcommand*\linerefr[2][\z@]{%
619   \def\@@linerefadd{\advance\c@linenumber#1}%
620   \expandafter\@setref\csname r@#2\endcsname
621   \@linerefadd{#2}%
622 }}
623
624 \newcommand*\@linerefadd[2]{\c@linenumber=#1\@@linerefadd\relax
625                               \thelinenumber}
```

## 7.4 Numbered quotation environments

7 The `{numquote}` and `{numquotation}` environments are like `{quote}` and `{quotation}`, except there will be line numbers.

10 An optional argument gives the number to count from. A star `*` (inside or outside the closing `}`) prevent the reset of the line numbers. Default is to count from one.

13 (v4.22: A local version using `\c@internallinenumber` might be useful, see subsection 7.2.)

```
626 \newcommand\quotelinenumbers
627   {\@ifstar\linenumbers{\@ifnextchar[\linenumbers{\linenumbers*}}}
628
```

```

629 \newdimen\quotelinumbersep
630 \quotelinumbersep=\linumbersep
631 \let\quotelinumberfont\linumberfont
632
633 \newcommand\numquotelist
634   {\leftlinenumbers
635     \linumbersep\quotelinumbersep
636     \let\linumberfont\quotelinumberfont
637     \addtolength{\linumbersep}{-\@totalleftmargin}%
638     \quotelinumbers
639   }
640
641 \newenvironment{numquote}   {\quote\numquotelist}\endquote}
642 \newenvironment{numquotation} {\quotation\numquotelist}\endquotation}
643 \newenvironment{numquote*}   {\quote\numquotelist*}\endquote}
644 \newenvironment{numquotation*} {\quotation\numquotelist*}\endquotation}

```

## 1 7.5 Frame around a paragraph

The `{bframe}` environment draws a frame around some text, across page breaks, if necessary.

- 4 This works only for plain text paragraphs, without special height lines. All lines must be `\baselineskip` apart, no display math.

```

645 \newenvironment{bframe}
646   {\par
647     \@tempdima\columnwidth
648     \advance\@tempdima 2\bframesep
649     \setbox\bframebox\hb@xt@\columnwidth{%
650       \hskip-\bframesep
651       \vrule\@width\bframerule\@height\baselineskip\@depth\bframesep
652       \advance\@tempdima-2\bframerule
653       \hskip\@tempdima
654       \vrule\@width\bframerule\@height\baselineskip\@depth\bframesep
655       \hskip-\bframesep
656     }%
657     \hbox{\hskip-\bframesep
658       \vrule\@width\@tempdima\@height\bframerule\@depth\z@}%
659     \nointerlineskip
660     \copy\bframebox
661     \nobreak
662     \kern-\baselineskip
663     \runninglinenumbers
664     \def\makeLineNumber{\copy\bframebox\hss}%
665   }
666   {\par
667     \ifdim\prevdepth=-1000pt \else
668       \kern-\prevdepth

```

```

669 \fi
670 \kern\bframesep
671 \nointerlineskip
672 \@tempdima\columnwidth
673 \advance\@tempdima 2\bframesep
674 \hbox{\hskip-\bframesep
675 \vrule\@width\@tempdima\@height\bframerule\@depth\z@}%
676 }
677
678 \newdimen\bframerule
679 \bframerule=\fboxrule
680
681 \newdimen\bframesep
682 \bframesep=\fboxsep
683
684 \newbox\bframebox

```

## 1 7.6 amsmath patches

(New v5.0) Patches `amsmath` to work with `lineno`. These patches used to be supplied by the `linenoamsmath` package. See `linenoamsmathdemo.pdf` for a demonstration. (/New v5.0)

(New v5.1) `lineno` tries to use L<sup>A</sup>T<sub>E</sub>X's hook management system to patch `amsmath`, so that the two packages may be loaded independently. This requires the October 2020 release of L<sup>A</sup>T<sub>E</sub>X. As a fallback for older releases, `lineno` tests whether `amsmath` had already been loaded (by testing for the presence of the `gather` command) and if so applies the patches; otherwise if `amsmath` has not been loaded, no patches are applied, and a warning is issued. (/New v5.1)

(New v5.2) Fix `lineno` to work with `amsmath`'s `\allowdisplaybreaks` option. A side effect is that now `\*` suppresses a line number on that line. This is because `\*` prohibits a page break after a given line, and `lineno` basically works by hijacking page breaks. It's probably not possible to fix this without losing the behaviour of `\*`. (/New v5.2)

```

685 \ifdefined\AddToHook
686 \def\LN@amsmath@patches#1{\AddToHook{package/amsmath/after}{#1}}
687 \else
688 \ifdefined\endgather
689 \def\LN@amsmath@patches#1{#1}
690 \else
691 \PackageWarningNoLine{lineno}%
692 {'amsmath' must be loaded before 'lineno' for patches to be applied}
693 \def\LN@amsmath@patches#1{\relax}
694 \fi

```

```

695 \fi
696
697 \@LN@amsmath@patches{
698   \newcommand*\@LN@amsmath@patch[1]{%
699     \cspreto{#1}{\linenomath}%
700     \cspreto{#1*}{\linenomath}%
701     \csappto{end#1}{\endlinenomath}%
702     \csappto{end#1*}{\endlinenomath}%
703   }
704   \newcount\@LN@amsmath@ams@eqpen
705   \cspreto{math@cr@}{%
706     %% Uncommenting the following 2 lines restores the line number on a line
707     %% ended with \\*, by making \\* act just like \\. This is probably
708     %% undesirable, however, so these lines are disabled.
709     % \global\@eqpen%
710     % \ifnum\dsprk@lvl <\z@ \interdisplaylinepenalty \else -\@getpen\dsprk@lvl \fi%
711     \advance\@eqpen\@LN@amsmath@ams@eqpen\relax%
712   }
713   \newcommand*\@LN@amsmath@patch@ams[1]{%
714     \cspreto{#1}{%
715       \linenomath%
716       \postdisplaypenalty=0%
717       \global\@LN@amsmath@ams@eqpen\interdisplaylinepenalty%
718     }%
719     \cspreto{#1*}{%
720       \linenomath%
721       \postdisplaypenalty=0%
722       \global\@LN@amsmath@ams@eqpen\interdisplaylinepenalty%
723     }%
724     \csappto{end#1}{%
725       \global\@LN@amsmath@ams@eqpen\z@%
726       \endlinenomath%
727     }%
728     \csappto{end#1*}{%
729       \global\@LN@amsmath@ams@eqpen\z@%
730       \endlinenomath%
731     }%
732   }
733   \@LN@amsmath@patch{equation}
734   \@LN@amsmath@patch@ams{multline}
735   \@LN@amsmath@patch@ams{gather}
736   \@LN@amsmath@patch@ams{align}
737   \@LN@amsmath@patch@ams{alignat}
738   \@LN@amsmath@patch@ams{flalign}
739   \let\@LN@amsmath@ams@mmeasure\mmeasure@
740   \def\mmeasure@#1{%
741     \global\@LN@amsmath@ams@eqpen\z@%
742     \begingroup%
743     \interdisplaylinepenalty=0%

```

```

744 \LN@amsmath@ams@mmeasure{#1\\}%
745 \endgroup%
746 \global\LN@amsmath@ams@eqpen\interdisplaylinepenalty%
747 }
748 }

```

## 1 8 Move \vadjust items (New v4.00)

This section completes reviving `\pagebreak`, `\nopagebreak`, `\vspace`, and the star and optional form of `\`. This was started in section 2.1 and resumed in section 2.4 and subsection 3.1. The problem was explained in section 2.1: `\vadjust` items come out at a bad position, and the  $\LaTeX$  commands named before work with `\vadjust` indeed. Our solution was sketched there as well.

According to the caveat in subsection 3.2 concerning `\ifLineNumbers`, the  $\LaTeX$  commands enumerated may go wrong if you switch line numbering inside or at the end of a paragraph.

### 10 8.1 Redefining \vadjust

`\vadjust` will temporarily be changed into the following command.

```

749 \def\PostponeVadjust#1{%
750 \global\let\vadjust\LN@@vadjust

```

This undoes a `\global\let\vadjust\PostponeVadjust` which will start each of the refined  $\LaTeX$  commands. The `\globals` are most probably superfluous. They might be useful should one `\vadjust` appear in a group starting after the change of `\vadjust` into `\PostponeVadjust`. (UL) Even the undoing may be superfluous, cf. discussion in section 8.2 below. (UL)

```

751 \vadjust{\penalty-\@Mppvacodepen}%
752 \g@addto@macro\LN@vadjustlist{#1\@lt}%
753 }
754 \let\LN@@vadjust\vadjust
755 \global\let\LN@vadjustlist\@empty
756 \global\let\LN@do@vadjusts\relax

```

These `\globals` are just to remind that all the changes of the strings after `\let` should be `\global` ( $\TeX$ book p. 301). `\LN@vadjustlist` collects the `\vadjust` items of a paragraph. `\PassVadjustList` tears one `\vadjust` item for the current line out of `\LN@vadjustlist` and puts it into `\LN@do@vadjusts`. The latter is encountered each line in `\MakeLineNo` (section 2.4), while those  $\LaTeX$  `\vadjust` commands will come rather rarely.

- 1 So I decided that `\@LN@do@vadjust` is `\relax` until a `\vadjust` item is wait-  
 ing. In the latter case, `\@LN@do@vadjusts` is turned into a list macro which  
 4 resets itself to `\relax` when the other contents have been placed in the verti-  
 cal list.—`\PassVadjustList` is invoked by the output routine (section 2.1),  
 so the `\box255` must be put back.

```

757 \def\PassVadjustList{%
758   \unvbox\@cclv
759   \expandafter \@LN@xnext \@LN@vadjustlist \@@
760                   \@tempa \@LN@vadjustlist
761   \ifx\@LN@do@vadjusts\relax
762     \gdef\@LN@do@vadjusts{\global\let\@LN@do@vadjusts\relax}%
763   \fi
764   \expandafter \g@addto@macro \expandafter \@LN@do@vadjusts
765     \expandafter {\@tempa}%
766 }

```

## 8.2 Redefining the L<sup>A</sup>T<sub>E</sub>X commands

- 7 Now we change `\pagebreak` etc. so that they use `\PostponeVadjust` in  
 place of `\vadjust`. We try to do this as independently as possible of  
 the implementation of the L<sup>A</sup>T<sub>E</sub>X commands to be redefined. Therefore,  
 10 we don't just copy macro definition code from any single implementa-  
 tion (say, latest L<sup>A</sup>T<sub>E</sub>X) and insert our changes, but attach a conditional  
`\global\let\vadjust\PostponeVadjust` to their left ends in a way which  
 13 should work rather independantly of their actual code. However, `\vadjust`  
 should be the primitive again after execution of the command. So the  
`\global\let...` may be used only if it's guaranteed that a `\vadjust` is  
 16 near.—(UL) Sure? In line numbering mode, probably each `\vadjust` com-  
 ing from a L<sup>A</sup>T<sub>E</sub>X command should be `\PostponeVadjust`. `\marginpars`  
 and floats seem to be the only cases which are not explicitly dealt with in  
 19 the present section. This would be a way to avoid `\@LN@nobreaktrue!` Of  
 course, the `\vadjusts` that the present package uses then must be replaced  
 by `\@LN@@vadjust`.—Maybe next time. (/UL)
- 22 The next command and something else will be added to the L<sup>A</sup>T<sub>E</sub>X com-  
 mands we are concerned with here.

```

767 \DeclareRobustCommand\@LN@changevadjust{%
768   \ifvmode\else\ifinner\else
769     \global\let\vadjust\PostponeVadjust
770   \fi\fi
771 }

```

(UL) What about math mode? Math display? Warn? (/UL)

1 \@tempa will now become a two place macro which adds first argu-  
 ment (single token), enclosed by \ifLineNumbers... \fi to the left of sec-  
 ond argument. As long as we need it, we can't use the star form of  
 4 \DeclareRobustCommand or the like, because AMS- $\LaTeX$  uses \@tempa for  
 \@ifstar. (New v4.41) And for the same reason, that \CheckCommand\* had  
 to be raised! (/New v4.41)

```
772 \CheckCommand*\@parboxrestore{\@arrayparboxrestore\let\\\@normalcr}
773
774 \def\@tempa#1#2{%
775   \expandafter \def \expandafter#2\expandafter{\expandafter
776     \ifLineNumbers\expandafter#1\expandafter\fi#2}%
777 }
```

7 (UL) This \ifLineNumber can be fooled by \linenumbers ahead etc. It  
 might be better to place a signal penalty in any case and let the output  
 routine decide what to do. (/UL)

10 We use the occasion to switch off linenumbers where they don't work  
 anyway and where we don't want them, especially in footnotes:

```
778 \@tempa\nolinenumbers\@arrayparboxrestore
```

We hope this suffices ... let's check one thing at least: [(New v4.41) see  
 13 \CheckCommand above (/New v4.41)]

Now for the main theme of the section. The next lines assume that  
 \vspace, \pagebreak, and \nopagebreak use \vadjust whenever they occur  
 16 outside vertical mode; moreover, that they don't directly read an argument.  
 Indeed \pagebreak and \nopagebreak first call something which tests for a  
 left bracket ahead, while \vspace first tests for a star.

```
779 \@tempa\@LN@changeadjust\vspace
780 \@tempa\@LN@changeadjust\pagebreak
781 \@tempa\@LN@changeadjust\nopagebreak
```

19 \\\, however, uses \vadjust only in star or optional form. We relax indepen-  
 dency of implementation in assuming that \@normalcr is the fragile version  
 of \\\ (and we use \@ifstar!). (Using a copy of \\\ would be safer, but an  
 22 ugly repetition of \protect.)

```
782 \protected\def\\\{%
783   \ifLineNumbers
784     \expandafter \@LN@cr
785   \else
786     \expandafter \@normalcr
787   \fi
```

```

788 }
789 \def\@LN@cr{%
790   \@ifstar{\@LN@changeadjust\@normalcr*}%
791           {\@ifnextchar[{\@LN@changeadjust\@normalcr}\@normalcr}%
792 }

```

1 Moreover we hope that `\newline` never leads to a `\adjust`, although names of some commands invoked by `\` contain `newline`. At last, this seems to have been OK since 1989 or even earlier.

4 Let's have a few tests. Testing `\pagebreak` and `\nopagebreak` would  
5 be too expensive here, but—oops!—we have just experienced a successful  
6 `\vspace*{.5\baselineskip}`. A `\[*[.5\baselineskip]`  
7 may look even more drastical, but this time we are happy about it. Note  
8 that the line numbers have moved with the lines. Without our changes, one  
9 line number would have “anticipated” the move of the next line, just as you  
10 can observe it now. (/New v4.00)

### 11 8.3 Reminder on obsolescence

12 (New v4.1) We have completed inclusion of the earlier extension packages  
13 `linenox0.sty`, `linenox1.sty`, `lnopatch.sty`, and `linenoamsmath`. If one  
14 of them is loaded, though, we produce an error message before something  
15 weird happens. We avoid `\newif` because the switchings occur so rarely.

```

793 \AtBeginDocument{%
794   \let\ifLN@obsolete\iffalse
795   \@ifpackageloaded{linenox0}{\let\ifLN@obsolete\iftrue}\relax
796   \@ifpackageloaded{linenox1}{\let\ifLN@obsolete\iftrue}\relax
797   \@ifpackageloaded{lnopatch}{\let\ifLN@obsolete\iftrue}\relax
798   \@ifpackageloaded{linenoamsmath}{\let\ifLN@obsolete\iftrue}\relax
799   \ifLN@obsolete
800     \PackageError{lineno}{Obsolete extension package(s)}{%
801       As of \fileversion, ‘lineno’ includes the functionality of \MessageBreak
802       ‘linenox0’, ‘linenox1’, ‘lnopatch’, and ‘linenoamsmath’; \MessageBreak
803       these packages are therefore obsolete and must not be loaded.}%
804   \fi
805 }

```

## 16 9 The final touch

17 There is one deadcycle for each line number.

```
806 \advance\maxdeadcycles 100
807
808 \endinput
```

## 10 The user commands 1

The user commands to turn on and off line numbering are 2

`\linenumbers` 3

Turn on line numbering in the current mode. 4

`\linenumbers*` 5

and reset the line number to 1. 6

`\linenumbers[⟨number⟩]` 7

and start with ⟨*number*⟩. 8

`\nolinenumbers` 9

Turn off line numbering. 10

`\runninglinenumbers*[⟨number⟩]` 11

Turn on `running` line numbers, with the same optional arguments as `\linenumbers`. The numbers are running through the text over page-breaks. When you turn numbering off and on again, the numbers will continue, except, of course, if you ask to reset or preset the counter. 12-15

`\pagewiselinenumbers` 16

Turn on `pagewise` line numbers. The lines on each page are numbered beginning with one at the first `pagewise` numbered line. 17-18

`\resetlinenumber[⟨number⟩]` 19

Reset [Set] the line number to 1 [⟨*number*⟩]. 20

`\setrunninglinenumbers` 21

Switch to `running` line number mode. Do *not* turn it on or off. 22

`\setpagewiselinenumbers` 23

Switch to `pagewise` line number mode. Do *not* turn it on or off. 24

1 `\switchlinenumbers*`

2 Causes margin switching in pagewise modes. With the star, put the  
3 line numbers on the inner margin.

4 `\leftlinenumbers*`

5 `\rightlinenumbers*`

6 Set the line numbers in the left/right margin. With the star this works  
7 for both modes of operation, without the star only for the currently  
8 selected mode.

9 `\runningpagewiselinenumbers`

10 When using the pagewise line number mode, do not subtract the page  
11 offset. This results in running line numbers again, but with the possibil-  
12 ity to switch margins. Be careful when doing line number referencing,  
13 this mode status must be the same while setting the paragraph and  
14 during references.

15 `\realpagewiselinenumbers`

16 Reverses the effect of `\runningpagewiselinenumbers`.

17 `\modulolinenumbers[⟨number⟩]`

18 Give a number only to lines which are multiples of [⟨number⟩].  
19 If ⟨number⟩ is not specified, the current value in the counter  
20 `linenumbermodulo` is retained. ⟨number⟩=1 turns this off without  
21 changing `linenumbermodulo`. The counter is initialized to 5.

22 `\modulolinenumbers*[⟨number⟩]`

23 Like `\modulolinenumbers`, the only difference being that the  
24 first line number after a `\linenumbers` (or `\runninglinenumbers`,  
25 `\pagewiselinenumbers`, `\quotelinenumbers`) is printed regard-  
26 less of the modulo—yet ‘1’ is printed only after (or ...) `\firstlinenumber{1}`. This also applies to the first line of a  
27 `{linenumbers}` or respective environment. See subsection 5.5 for an-  
28 other explanation. The behaviour may be unsatisfactory with pagewise  
29 line-numbering.  
30

31 `\firstlinenumber`

32 `\firstlinenumber{⟨filino⟩}` brings about that (after it) line num-  
33 bers less than ⟨filino⟩ do *not* appear in the margin. Moreover,  
34 with `\modulolinenumbers[⟨number⟩]`, just the line numbers which

are  $\langle filino \rangle$  plus a multiple of  $\langle number \rangle$  are printed.—If you had `\firstlinenumber{ $\langle pos \rangle$ }` with some  $\langle pos \rangle > 0$  and want to switch to printing multiples of, e.g., 4, you best do `\modulolinenumbers[4]` and `\firstlinenumber{0}`. (See subsection 5.5 for technical details.)

`\linenumberdisplaymath` 5

Number the lines of a display math in a `{linenomath}` environment, but do not in a `{linenomath*}` environment. This is used by the package option `[mathlines]`.

`\nolinenumberdisplaymath` 9

Do not Number the lines of a display math in a `{linenomath}` environment, but do in a `{linenomath*}` environment. This is the default.

`\linelabel` 12

Set a `\linelabel{ $\langle foo \rangle$ }` to the line number where this commands is in. Refer to it with the L<sup>A</sup>T<sub>E</sub>X referencing commands `\ref{ $\langle foo \rangle$ }` and `\pageref{ $\langle foo \rangle$ }`.

The commands can be used globally, locally within groups or as environments. It is important to know that they take action only when the `\par` is executed. The `\end{ $\langle mode \rangle$ linenumbers}` commands provide a `\par`. Examples:

```
{\linenumbers  $\langle text \rangle$  \par} 20
```

```
\begin{linenumbers} 21
```

```
 $\langle text \rangle$  22
```

```
\end{linenumbers} 23
```

```
 $\langle paragraph \rangle$  {\linenumbers\par} 24
```

```
\linenumbers 25
```

```
 $\langle text \rangle$  \par 26
```

```
\nolinenumbers 27
```

```
\linenumbers 28
```

```
 $\langle paragraph \rangle$  {\nolinenumbers\par} 29
```

(New v4.00) However, the examples containing  $\langle paragraph \rangle$  show what you should *not* do, at least if you use `\pagebreak`, `\nopagebreak`, `\vspace`, `\*` or `\\[ $\langle space \rangle$ ]`—cf. section 8.

1 The same care should be applied to the “wizard” devices `\ifLineNumbers`  
 2 (subsection 3.2) and `\PostponeVadjust` (section 8.1). (/New v4.00)  
 3 (New v4.11) Oh, and the commands and environments of section s:Xt are  
 4 missing. Sorry, I am in a hurry now. May be next time.—And the environ-  
 5 ments `{linenomath}` and `{linenomath*}` should get an own paragraph. In  
 6 short, each math display, equation, or `{eqnarray}` should be “wrapped” in  
 7 one of `{linenomath}` and `{linenomath*}`.

## 8 10.1 Customization hooks

9 There are several hooks to customize the appearance of the line numbers,  
 10 and some low level hooks for special effects.

11 `\thelinenum`

12 This macro should give the representation of the line number in the  
 13  $\LaTeX$ -counter `linenumber`. The default is provided by  $\LaTeX$ :

14 `\arabic{linenumber}`

15 `\makeLineNumberLeft`

16 This macro is used to attach a line number to the left of the text page.  
 17 This macro should fill an `\hbox` to `Opt` which will be placed at the  
 18 left margin of the page, with the reference point aligned to the line to  
 19 which it should give a number. Please use the macro `\LineNumber` to  
 20 refer to the line number.

21 The default definition is

22 `\hss\linenumberfont\LineNumber\hskip\linenumbersep`

23 `\makeLineNumberRight`

24 Like `\makeLineNumberLeft`, but for line numbers on the right margin.

25 The default definition is

26 `\linenumberfont\hskip\linenumbersep\hskip\columnwidth`

27 `\hbox to\linenumberwidth{\hss\LineNumber}\hss`

28 `\linenumberfont`

29 This macro is initialized to

30 `\normalfont\tiny\sffamily`

<code>\linenumbersep</code>	1
This dimension register sets the separation of the line number to the text. Default value is <code>10pt</code> .	2 3
<code>\linenumberwidth</code>	4
This dimension register sets the width of the line number box on the right margin. The distance of the right edge of the text to the right edge of the line number is <code>\linenumbersep + \linenumberwidth</code> . The default value is <code>10pt</code> .	5 6 7 8
<code>\theLineNumber</code> (for wizards)	9
This macro is called for printing a <code>\newlabel</code> entry to the aux-file. Its definition depends on the mode. For running line numbers it's just <code>\thelinenumber</code> , while in pagewise mode, the page offset subtraction is done in here.	10 11 12 13
<code>\makeLineNumber</code> (for wizards)	14
This macro produces the line numbers. The definition depends on the mode. In the running line numbers mode it just expands <code>\makeLineNumberLeft</code> .	15 16 17
<code>\LineNumber</code> (for wizards)	18
This macro is called by <code>\makeLineNumber</code> to typeset the line number. This hook is changed by the modulo mechanism and by <code>\firstlinenumber</code> .	19 20 21