

The `amsmath` package

Frank Mittelbach Rainer Schöpf Michael Downes
David M. Jones

Version 2.14, 2013/01/14

1 Introduction

A \LaTeX package named `amstex` was created in 1988–1989 by adapting `amstex.tex` for use within \LaTeX . The `amsmath` package is the successor of the `amstex` package. It was substantially overhauled to integrate it with $\text{\LaTeX}2\epsilon$, which arrived on the scene in 1994. It provides more or less the same features, but there are quite a few organizational differences as well as some new features and bug fixes. For example, the `amstex` package automatically loaded the `amsfonts` package, but the `amsmath` package does not. At the present time (November 1999) user-level documentation of the commands provided here is found in the AMSmath Users' Guide, `amsldoc.tex`.

Standard file identification.

```
1 \NeedsTeXFormat{LaTeX2e}% LaTeX 2.09 can't be used (nor non-LaTeX)
2 [1994/12/01]% LaTeX date must be December 1994 or later
3 \ProvidesPackage{amsmath}[2013/01/14 v2.14 AMS math features]
```

2 Catcode defenses

Some packages change the catcode of characters that are essential in low-level \TeX syntax. Any package that does so does not qualify as a PWWO package (“Plays Well With Others”) because it can cause other packages to fail if they are loaded later. \LaTeX is partly to blame for this because it fails to provide adequate built-in safeguards in the package loading mechanisms. In the absence of such safeguards, we will provide them here.

```
4 \edef\@temp{\catcode 96=\number\catcode 96 }
5 \catcode\string ``=12
6 \def\do#1{\catcode\number`#1=\number\catcode`#1}
7 \edef\@temp{%
8   \noexpand\AtEndOfPackage{%
9     \@temp
10    \do\" \do\' \do\(\do\)\do*\do+\do\,\do-\do\.%
11    \do\/\do\<\do\=\do\>\do\[ \do\] \do\^ \do\_ \relax
12  }%
13 }
14 \@temp
15 \def\do#1{\catcode\number`#1=12 }
16 \do\" \do\' \do\(\do\)\do*\do+\do\,\do-\do\.%
17 \do\/\do\<\do\=\do\>\do\[ \do\]
18 \catcode``=7 \catcode\_ =8
```

3 Declare some options

Handling of limits on integrals, sums, operatornames.

```
19 \DeclareOption{intllimits}{\let\ilimits@\displaylimits}
20 \DeclareOption{nointllimits}{\let\ilimits@\nolimits}
```

```

21 \DeclareOption{sumlimits}{\let\slimits@\displaylimits}
22 \DeclareOption{nosumlimits}{\let\slimits@\nolimits}
23 \DeclareOption{namelimits}{\PassOptionsToPackage{namelimits}{amsopn}}
24 \DeclareOption{nonamelimits}{%
25   \PassOptionsToPackage{nonamelimits}{amsopn}}

```

The following two switches might have been defined already by the documentclass, but it doesn't hurt to re-execute the `\newif`'s.

```

26 \newif\ifctagsplit@
27 \newif\iftagsleft@

```

Right or left placement of equation numbers.

```

28 \DeclareOption{leqno}{\tagsleft@true}
29 \DeclareOption{reqno}{\tagsleft@false}
30 \DeclareOption{centertags}{\ctagsplit@true}
31 \DeclareOption{tbtags}{\ctagsplit@false}

```

The `cmex10` option is an escape hatch for people who don't happen to have sizes 7–9 of the `cmex` fonts available to them yet. (Strictly speaking they are considered part of a minimum \LaTeX distribution now, i.e., all \LaTeX 2_ε users should have them, without needing to get the `AMSFonTS` distrib.)

```

32 \DeclareOption{cmex10}{%
33   \ifnum\cmex@opt=\@ne \def\cmex@opt{0}%
34   \else \def\cmex@opt{10}\fi
35 }

```

To help things work out better with various package loading orders of `amsmath` and `amsfonts`, we establish a variable to communicate the status of the `cmex` font definition. If the `amsfonts` package was loaded first this variable might be already defined, in which case we want to preserve its value.

```

36 \@ifundefined{cmex@opt}{\def\cmex@opt{7}}{}

```

4 Flush-left equations [DMJ]

The left margin of math environments is controlled by `\@mathmargin`. This can be set to `\@centering` to implement the default behaviour, i.e., centered equations, and to something else to implement the flushleft style.

In theory, all that's needed to activate the flushleft mode in the AMS document classes is something like this:

```

\DeclareOption{fleqn}{%
  \AtBeginDocument{\@mathmargin30pt\relax}%
}

```

(In fact, unless the document class wants to specify the `\@mathmargin`, it doesn't need to do anything with the `fleqn` option.)

```

37 \newif\if@fleqn
38 %
39 \newskip\@mathmargin
40 \@mathmargin\@centering
41 %
42 \DeclareOption{fleqn}{%
43   \@fleqntrue
44   \@mathmargin = -1sp
45   \let\mathindent=\@mathmargin
46   \AtBeginDocument{%
47     \ifdim\@mathmargin= -1sp
48       \@mathmargin\leftmargini minus\leftmargini
49     \fi
50   }%
51 }

```

DMJ: This ensures that `\@mathmargin` is given some sort of sensible default if the class doesn't specify one, while still allowing a user to override the default value in their document preamble. (Incidentally, I'm initializing `\@mathmargin` to `\leftmargini` for compatibility with `fleqn.clo`, but I'm not at all convinced that's the right thing to do.)

The next question is what happens when `amsmath` is used with one of the standard classes. Unfortunately, \LaTeX implements `fleqn` somewhat clumsily; instead of parameterizing the definitions of the math structures (as I've attempted to do here), `fleqn.clo` declares a `\dimen \mathindent` that is much like my `\@mathmargin` and then redefines `\[`, `\]`, `\equation`, and `\eqnarray`. This means that things could get rather messy in 2.09 compatibility mode, since `fleqn.clo` might be loaded after `amsmath.sty`, which could cause a real mess.

[mjd,1999/07/07]: Let `\mathindent = \@mathmargin` as envisioned by DMJ. Compatibility-mode documents will all use the `amstex` package, not `amsmath`. There is a remote chance of a problem if someone makes an assignment to `\mathindent` in a way that implicitly assumes it is a `\dimen` register (inasmuch as it has now become a `\skip` register), and the string "plus" follows in the input stream, but if someone's document croaks in that way, I think they will just have to bite the bullet and fix it. The alternative is to penalize a lot of other users with a known handicap.

```
52 \DeclareOption{?}{}
```

```
53 \ExecuteOptions{nointlimits,sumlimits,namelimits,centertags}
```

The `\par` after `\ProcessOptions` is to ensure the correct line number on screen if an error occurs during option processing; otherwise the lookahead for a `*` option would result in \TeX showing the following line instead.

```
54 \ProcessOptions\par
```

```
55 \@ifpackagewith{amsmath}{?}{%
```

```
56   \typeout{^^J%
```

```
57 Documentation for the amsmath package is found in amsldoc.dvi^^J%
```

```
58 (or .pdf or .tex).^^J%
```

```
59 ^^J%
```

```
60 See also http://www.ams.org/tex/amslatex.html.^^J%
```

```
61 ^^J%
```

```
62 Note: Using the first edition of The LaTeX Companion (1994) without^^J%
```

```
63 errata as a guide for amsmath use is not recommended.^^J%
```

```
64 }%
```

```
65 }{%
```

```
66   \typeout{%
```

```
67 For additional information on amsmath, use the \lq ?\rq\space option.%
```

```
68 }%
```

```
69 }
```

Processing to handle the `cmex10` option is a little tricky because of different possible loading orders for `amsmath` and `amsfonts`. The package `amsmath` sets the `\cmex@opt` flag to 7 or 10, and the package `amsfonts` sets the flag to 1 or 0.

```
70 \ifnum\cmex@opt=7 \relax
```

```
71   \DeclareFontShape{OMX}{cmex}{m}{n}{%
```

```
72     <-8>cmex7<8>cmex8<9>cmex9%
```

```
73     <10><10.95><12><14.4><17.28><20.74><24.88>cmex10%
```

```
74   }{ }%
```

```
75   \expandafter\let\csname OMX/cmex/m/n/10\endcsname\relax
```

```
76 \else
```

```
77   \ifnum\cmex@opt=\z@ % need to override cmex7 fontdef from amsfonts
```

Force reloading of the `OMX/cmex` font definition file.

```
78   \begingroup
```

```

79 \fontencoding{OMX}\fontfamily{cmex}%
80 \expandafter\let\csname OMX+cmex\endcsname\relax
81 \try@load@fontshape
82 \endgroup

```

The `cmex10` font gets special preload handling in the building of the \LaTeX format file, need an extra bit here to work around that.

```

83 \expandafter\let\csname OMX/cmex/m/n/10\endcsname\relax
84 \def\cmex@opt{10}%
85 \fi
86 \fi

```

5 Call some other packages

The `amstext` package provides the `\text` command. The `amsbsy` package provides `\boldsymbol` and `\pmb`. (Since 1997 it is usually better to use the `bm` package instead; but I think we have to keep `amsbsy` here for backward compatibility [mjd,1999/11/19].) The `amsopn` package provides `\DeclareMathOperator`.

```

87 \RequirePackage{amstext}[1995/01/25]
88 \RequirePackage{amsbsy}[1995/01/20]
89 \RequirePackage{amsopn}[1995/01/20]

```

6 Miscellaneous

`\ams@newcommand` Where `stix` and `amsmath` define the same control sequences, we want to avoid inadvertently overriding `stix`'s definitions. If `stix` is loaded before `amsmath`, the following conditional takes care of the problem. There is similar code in the `stix` package in case `amsmath` is loaded first.

```

90 \@ifpackageloaded{stix}{%
91 \let\ams@newcommand\providecommand
92 \let\ams@renewcommand\providecommand
93 \let\ams@def\providecommand
94 \let\ams@DeclareRobustCommand\providecommand
95 }{%
96 \let\ams@newcommand\newcommand
97 \let\ams@renewcommand\renewcommand
98 \let\ams@def\def
99 \let\ams@DeclareRobustCommand\DeclareRobustCommand
100 }

```

`\@amsmath@err` Defining this error function saves main mem.

```
101 \def\@amsmath@err{\PackageError{amsmath}}
```

`\AmS` The `\AmS` prefix can be used to construct the combination `\AmS- \LaTeX` .

```

102 \providecommand{\AmS}{\protect\AmSfont
103 A\kern-.1667em\lower.5ex\hbox{M}\kern-.125emS}}

```

In `\AmSfont` we call `cmsy` directly in lieu of trying to access it through the math fonts setup (e.g. `\the\textfont2`) because math fonts can't be relied on to be properly set up if we are not inside a math formula. This means that if this command is used in a document where CM fonts are not wanted, then a font substitution will need to be declared, e.g.:

```
\DeclareFontShape{OMS}{cmsy}{m}{n}{ <-> sub * xxx/m/n }{ }
```

where `xxx` is some alternate font family. Taking the first letter of `\f@series` will produce `b` or `m` for the most common values (`b`, `bx`, `m`). It may produce nonsense for more unusual values of `\f@series`, so for safety's sake we have an additional

`\if` test. We want to avoid setting the series to `bx` because in a standard L^AT_EX installation the combination `cmsy/bx/n` does not have a font definition, and the user would get a font substitution warning on screen.

```
104 \newcommand{\AmSfont}{%
105   \usefont{OMS}{cmsy}{\if\@xp\@car\f@series\@nil bb\else m\fi}{n}}
```

`\@mathmeasure` The function `\@mathmeasure` takes three arguments; the third arg is typeset as a math formula in an hbox, using arg #2 as the mathstyle, and the result is left in the box named by the first arg. It is assumed that we are already in math mode, so we can turn off `\everymath` (in particular, `\check@mathfonts`).

```
106 \def\@mathmeasure#1#2#3{\setbox#1\hbox{\frozen@everymath\@emptytoks
107   \m@th$#2#3$}}
```

The `\inf@bad` constant is for testing overfull boxes.

```
108 \ifundefined{inf@bad}{%
109   \newcount\inf@bad \inf@bad=1000000 \relax
110 }{}
```

6.1 Math spacing commands

Here we fill in some gaps in the set of spacing commands, and make them all work equally well in or out of math. We want all these commands to be robust but declaring them all with `\DeclareRobustCommand` uses up a control sequence name per command; to avoid this, we define a common command `\tmspace` (text-or-math space) which carries the robustness burden for all of them. The standard `\relax` before the `\ifmmode` is not necessary because of the `\protect` added by `\DeclareRobustCommand`.

```
111 \DeclareRobustCommand{\tmspace}[3]{%
112   \ifmmode\mskip#1#2\else\kern#1#3\fi\relax}
113 \renewcommand{\,}{\tmspace+\thinmuskip{.1667em}}
114 \let\thinspace\,
115 \renewcommand{\!}{\tmspace-\thinmuskip{.1667em}}
116 \let\negthinspace\!
117 \renewcommand{\:}{\tmspace+\medmuskip{.2222em}}
118 \let\medspace\:
119 \newcommand{\negmedspace}{\tmspace-\medmuskip{.2222em}}
120 \renewcommand{\;}{\tmspace+\thickmuskip{.2777em}}
121 \let\thickspace\;
122 \newcommand{\negthickspace}{\tmspace-\thickmuskip{.2777em}}
```

`\mspace` And while we're at it, why don't we provide an equivalent of `\hspace` for math mode use. This allows use of mu units in (for example) constructing compound math symbols.

```
123 \newcommand{\mspace}[1]{\mskip#1\relax}
```

6.2 Vertical bar symbols

Add left/right specific versions of `\vert`, `\Vert`. Don't assume the delimiter codes are the CM defaults.

```
124 \def\@tempa#1#2\@nil{%
125   \ifx\delimiter#1\@tempcnta#2\relax\else\@tempcnta\z@\fi
126 }
127 \@xp\@tempa\vert\@empty\@nil
128 \ifnum\@tempcnta>\z@
129   \advance\@tempcnta "4000000
130   \xdef\lvert{\delimiter\number\@tempcnta\space }
131   \advance\@tempcnta "1000000
132   \xdef\rvert{\delimiter\number\@tempcnta\space }
133 \else
```

```

134 \ifx\@@undefined\lvert
135   % Fall back to cmex encoding since we don't know what else to do.
136   \DeclareMathDelimiter{\lvert}
137     {\mathopen}{symbols}{"6A}{largesymbols}{"0C}
138   \DeclareMathDelimiter{\rvert}
139     {\mathclose}{symbols}{"6A}{largesymbols}{"0C}
140 \fi
141 \fi
142 \@xp\@tempa\Vert\@empty\@nil
143 \ifnum\@tempcnta>\z@
144   \advance\@tempcnta "4000000
145   \xdef\lVert{\delimiter\number\@tempcnta\space }
146   \advance\@tempcnta "1000000
147   \xdef\rVert{\delimiter\number\@tempcnta\space }
148 \else
149   \ifx\@@undefined\lVert
150     \DeclareMathDelimiter{\lVert}
151       {\mathopen}{symbols}{"6B}{largesymbols}{"0D}
152     \DeclareMathDelimiter{\rVert}
153       {\mathclose}{symbols}{"6B}{largesymbols}{"0D}
154   \fi
155 \fi

```

6.3 Fractions

Bury the generalized fraction primitives `\over`, `\atop`, etc., because of their bizarre syntax, which is decidedly out of place in a \LaTeX document.

```

156 \@saveprimitive\over\@@over
157 \@saveprimitive\atop\@@atop
158 \@saveprimitive\above\@@above
159 \@saveprimitive\overwithdelims\@@overwithdelims
160 \@saveprimitive\atopwithdelims\@@atopwithdelims
161 \@saveprimitive\abovewithdelims\@@abovewithdelims

```

If someone insists on using `\over`, give a warning the first time and then resurrect the old definition. *Laissez-faire* policy.

```

162 \DeclareRobustCommand{\primfrac}[1]{%
163   \PackageWarning{amsmath}{%
164     Foreign command \@backslashchar#1;\MessageBreak
165     \protect\frac\space or \protect\genfrac\space should be used instead%
166     \MessageBreak
167   }
168   \global\@xp\let\csname#1\@xp\endcsname\csname @@#1\endcsname
169   \csname#1\endcsname
170 }
171 \renewcommand{\over}{\primfrac{over}}
172 \renewcommand{\atop}{\primfrac{atop}}
173 \renewcommand{\above}{\primfrac{above}}
174 \renewcommand{\overwithdelims}{\primfrac{overwithdelims}}
175 \renewcommand{\atopwithdelims}{\primfrac{atopwithdelims}}
176 \renewcommand{\abovewithdelims}{\primfrac{abovewithdelims}}

```

`\frac` calls `\@@over` directly instead of via `\genfrac`, for better speed because it is so common. `\tfrac` and `\dfrac` are abbreviations for some commonly needed mathstyle overrides. To conserve csnames we avoid making `\dfrac` and `\tfrac` robust (`\genfrac` is itself robust).

```

177 \DeclareRobustCommand{\frac}[2]{\begingroup#1\endgroup\@@over#2}}
178 \newcommand{\dfrac}{\genfrac{}{}{0}{}{}}
179 \newcommand{\tfrac}{\genfrac{}{}{1}{}{}}

```

The `\binom` command for binomial notation works like `\frac` and has similar variants. Note that we do not use `\z@` in `\dbinom` and `\tbinom` because they are not top-level robust like `\binom`, and so the `\z@` with the potentially problematic `@` character would become visible when writing one of those commands to a `.toc` file.

```
180 \DeclareRobustCommand{\binom}{\genfrac{}{}{}{}{}}
181 \newcommand{\dbinom}{\genfrac{}{}{}{}{}}
182 \newcommand{\tbinom}{\genfrac{}{}{}{}{}}
```

`\genfrac` This command provides access to \TeX 's generalized fraction primitives. Args: #1 left delim, #2 right delim, #3 line thickness, #4 mathstyle override, #5 numerator, #6 denominator. But we only read the first four args at first, in order to give us a moment to select the proper generalized fraction primitive. Any of those four args could be empty, and when empty the obvious defaults are selected (no delimiters, default line thickness (normally .4pt), and no mathstyle override).

```
183 \DeclareRobustCommand{\genfrac}[4]{%
184   \def\@tempa{#1#2}%
185   \edef\@tempb{\@nx\@genfrac\@mathstyle{#4}%
186     \csname @@\ifx @#3@over\else above\fi
187     \ifx \@tempa\@empty \else withdelims\fi\endcsname}
188   \@tempb{#1#2#3}}
```

`\@genfrac` takes the preceding arguments and reads the numerator and denominator. Note that there's no convenient way to make the numerator and denominator *contents* displaystyle, through this interface.

Args: #1 mathstyle, #2 fraction primitive, #3 delimiters and rule thickness, #4 numerator, #5 denominator.

```
189 \def\@genfrac#1#2#3#4#5{{#1{\begingroup#4\endgroup#2#3\relax#5}}}
```

Empty mathstyle arg: no change; 0 = displaystyle, 1 = textstyle, 2 = scriptstyle, 3 = scriptscriptstyle.

```
190 \def\@mathstyle#1{%
191   \ifx\@empty#1\@empty\relax
192   \else\ifcase#1\displaystyle % case 0
193     \or\textstyle\or\scriptstyle\else\scriptscriptstyle\fi\fi}
```

6.4 Sums and Integrals

Default value for sum limits is `\displaylimits`, see option 'nosumlimits'.

We redefine all the cumulative operator symbols to use `\slimits@` so that switching between `\displaylimits` and `\nolimits` can be controlled by package options. Also add `\DOTSB` for the benefit of the dots lookahead. But we'd better make sure `\coprod` and the others are simple mathchars; if not, the attempted changes will probably fail miserably.

```
194 \begingroup
195 \edef\@tempa{\string\mathchar"}
196 \def\@tempb#1"#2\@nil{#1"}
197 \edef\@tempc{\expandafter\@tempb\meaning\coprod "\@nil}
198 \ifx\@tempa\@tempc
199   \global\let\coprod@\coprod
200   \gdef\coprod{\DOTSB\coprod@\slimits@}
201   \global\let\bigvee@\bigvee
202   \gdef\bigvee{\DOTSB\bigvee@\slimits@}
203   \global\let\bigwedge@\bigwedge
204   \gdef\bigwedge{\DOTSB\bigwedge@\slimits@}
205   \global\let\biguplus@\biguplus
206   \gdef\biguplus{\DOTSB\biguplus@\slimits@}
```

```

207 \global\let\bigcap@bigcap
208 \gdef\bigcap{\DOTSB\bigcap@\slimits@}
209 \global\let\bigcup@bigcup
210 \gdef\bigcup{\DOTSB\bigcup@\slimits@}
211 \global\let\prod@prod
212 \gdef\prod{\DOTSB\prod@\slimits@}
213 \global\let\sum@sum
214 \gdef\sum{\DOTSB\sum@\slimits@}
215 \global\let\bigotimes@bigotimes
216 \gdef\bigotimes{\DOTSB\bigotimes@\slimits@}
217 \global\let\bigoplus@bigoplus
218 \gdef\bigoplus{\DOTSB\bigoplus@\slimits@}
219 \global\let\bigodot@bigodot
220 \gdef\bigodot{\DOTSB\bigodot@\slimits@}
221 \global\let\bigsqcup@bigsqcup
222 \gdef\bigsqcup{\DOTSB\bigsqcup@\slimits@}
223 \fi
224 \endgroup

```

6.5 Roots and radicals

This root stuff needs syntax work and implementation work. Surely something more compact can be done?? [mjd, 1994/09/05]

```

225 \newcommand{\leftroot}{\@amsmath@err{\Invalid@@\leftroot}\@eha}
226 \newcommand{\uproot}{\@amsmath@err{\Invalid@@\uproot}\@eha}
227 \newcount\uproot@
228 \newcount\leftroot@
229 \renewcommand{\root}{\relaxnext@
230 \DN@{\ifx\@let@token\uproot\let\next@nextii@else
231 \ifx\@let@token\leftroot\let\next@nextiii@else
232 \let\next@plainroot@\fi\fi\next@}%
233 \def\nextii@uproot##1{\uproot@##1\relax\FN@nextiv@}%
234 \def\nextiv@{\ifx\@let@token@sptoken\DN@.\{\FN@nextv@}\else
235 \DN@.\{\FN@nextv@}\fi\next@.}%
236 \def\nextv@{\ifx\@let@token\leftroot\let\next@nextvi@else
237 \let\next@plainroot@\fi\next@}%
238 \def\nextvi@leftroot##1{\leftroot@##1\relax\plainroot@}%
239 \def\nextiii@leftroot##1{\leftroot@##1\relax\FN@nextvii@}%
240 \def\nextvii@{\ifx\@let@token@sptoken
241 \DN@.\{\FN@nextviii@}\else
242 \DN@.\{\FN@nextviii@}\fi\next@.}%
243 \def\nextviii@{\ifx\@let@token\uproot\let\next@nextix@else
244 \let\next@plainroot@\fi\next@}%
245 \def\nextix@uproot##1{\uproot@##1\relax\plainroot@}%
246 \bgroup\uproot@z@\leftroot@z@\FN@next@}
247 \def\plainroot@#1\of#2{\setbox\rootbox\hbox{%
248 $\m@th\scriptscriptstyle{#1}$}%
249 \mathchoice{\r@@t\displaystyle{#2}}{\r@@t\textstyle{#2}}
250 {\r@@t\scriptstyle{#2}}{\r@@t\scriptscriptstyle{#2}}\egroup}

```

Name change from `\@sqrt` to `\sqrtsign` happened in the 1995/12/01 release of L^AT_EX. If we were to assume that `\sqrtsign` is defined then someone with the 1995/06/01 release of L^AT_EX would have trouble using this package.

```

251 \@ifundefined{sqrtsign}{\let\sqrtsign\@sqrt}{ }
252 \def\r@@t#1#2{\setboxz@h{$\m@th#1\sqrtsign{#2}$}%
253 \dimen@ht\z@\advance\dimen@-\dp\z@
254 \setbox@one\hbox{$\m@th#1\mskip\uproot@ mu$}%
255 \advance\dimen@ by1.667\wd@one
256 \mkern-\leftroot@ mu\mkern5mu\raise.6\dimen@\copy\rootbox
257 \mkern-10mu\mkern\leftroot@ mu\boxz@}

```


6.6 Et cetera

Specific names for the variant italic cap Greek letters are not defined by L^AT_EX. If no preceding package defined these, we will define them now.

```

258 \ifundefined{varGamma}{%
259   \DeclareMathSymbol{\varGamma}{\mathord}{letters}{"00}
260   \DeclareMathSymbol{\varDelta}{\mathord}{letters}{"01}
261   \DeclareMathSymbol{\varTheta}{\mathord}{letters}{"02}
262   \DeclareMathSymbol{\varLambda}{\mathord}{letters}{"03}
263   \DeclareMathSymbol{\varXi}{\mathord}{letters}{"04}
264   \DeclareMathSymbol{\varPi}{\mathord}{letters}{"05}
265   \DeclareMathSymbol{\varSigma}{\mathord}{letters}{"06}
266   \DeclareMathSymbol{\varUpsilon}{\mathord}{letters}{"07}
267   \DeclareMathSymbol{\varPhi}{\mathord}{letters}{"08}
268   \DeclareMathSymbol{\varPsi}{\mathord}{letters}{"09}
269   \DeclareMathSymbol{\varOmega}{\mathord}{letters}{"0A}
270 }{}
```

$\mathcal{M}\mathcal{S}\text{-}\text{T}_{\text{E}}\text{X}$ redefines `\overline` as shown here, for reasons that are probably less important in L^AT_EX: Make it read its argument as a macro argument rather than a “math field” (*The T_EXbook*, Chapter 26), to avoid problems when something that is apparently a single symbol is actually a non-simple macro (e.g., `\dag`) and is given as a single-token argument without enclosing braces.

```

271 \saveprimitive\overline\@@overline
272 \DeclareRobustCommand{\overline}[1]{\@@overline{#1}}
```

The `\boxed` command is specifically intended to put a box around an equation or piece of an equation. (Not including the equation number.) This isn’t trivial for end-users to do it properly with `\fbox` so we provide a command for them.

```

273 \newcommand{\boxed}[1]{\fbox{\m@th$\displaystyle#1$}}
274 \newcommand{\implies}{\DOTSB\;\Longrightarrow\;}
275 \newcommand{\impliedby}{\DOTSB\;\Longleftarrow\;}
276 \def\And{\DOTSB\;\mathchar"3026 \;}

```

The command `\nobreakdash` is designed only for use before a hyphen or dash (–, --, or ---). Setting the hyphen in a box and then unboxing it means that the normal penalty will not be added after it—and if the penalty is not there a break will not be taken (unless an explicit penalty or glue follows, thus the final `\nobreak`).

```

277 \newcommand{\nobreakdash}{\leavevmode
278   \toks@{\emptytoks} \def\@tempa##1{\toks@{\xp{\the\toks@-}\FN@next@}%
279   \DN@{\ifx\@let@token-\@xp\@tempa
280     \else\setboxz@h{\the\toks@\nobreak}\unhbox\z@\fi}%
281   \FN@next@
282 }
```

`\colon` is for a colon in math that resembles a text colon: small space on the left, larger space on the right. The `:` character by itself is treated as a `\mathrel` i.e. large, equal spacing on both sides.

```

283 \renewcommand{\colon}{\nobreak\mskip2mu\mathpunct{}\nonscript
284   \mkern-\thinmuskip{:}\mskip6mu\plus1mu\relax}
```

7 Ellipsis dots

We can’t use `\newif` for `\ifgtest@` because we want to include `\global` in the definitions of `\gtest@true` and `\gtest@false`.

```

285 \let\ifgtest@iffalse % initial value
286 \def\gtest@true{\global\let\ifgtest@iftrue}
```

```

287 \def\gtest@false{\global\let\ifgtest@\iffalse}
288 \let\DOTSI\relax
289 \let\DOTSB\relax
290 \let\DOTSX\relax
291 {\uccode`7=`\ \uccode`8=`m \uccode`9=`a \uccode`0=`t \uccode`!=`h
292 \uppercase{%
293 \gdef\math@#1#2#3#4#5#6\math@{\gtest@false\ifx 7#1\ifx 8#2%
294 \ifx 9#3\ifx 0#4\ifx !#5\edef\meaning@{#6}\gtest@true
295 \fi\fi\fi\fi\fi}}
296 {\uccode`7=`c \uccode`8=`h \uccode`9=`\ "
297 \uppercase{\gdef\mathch@#1#2#3#4#5#6\mathch@{\gtest@false
298 \ifx 7#1\ifx 8#2\ifx 9#5\gtest@true\edef\meaning@{9#6}\fi\fi\fi}}
299 \newcount\classnum@
300 \def\getmathch@#1.#2\getmathch@{\classnum@#1 \divide\classnum@4096
301 \ifcase\number\classnum@\or\or\gdef\thedots@{\dotsb@}\or
302 \gdef\thedots@{\dotsb@}\fi}
303 {\uccode`4=`b \uccode`5=`i \uccode`6=`n
304 \uppercase{\gdef\mathbin@#1#2#3{\relaxnext@
305 \def\nextii@##1\mathbin@{\ifx\@sptoken\@let@token\gtest@true\fi}%
306 \gtest@false\DN@##1\mathbin@{}}%
307 \ifx 4#1\ifx 5#2\ifx 6#3\DN@{\FN@\nextii@}\fi\fi\fi\next@}}
308 {\uccode`4=`r \uccode`5=`e \uccode`6=`l
309 \uppercase{\gdef\mathrel@#1#2#3{\relaxnext@
310 \def\nextii@##1\mathrel@{\ifx\@sptoken\@let@token\gtest@true\fi}%
311 \gtest@false\DN@##1\mathrel@{}}%
312 \ifx 4#1\ifx 5#2\ifx 6#3\DN@{\FN@\nextii@}\fi\fi\fi\next@}}
313 {\uccode`5=`m \uccode`6=`a \uccode`7=`c
314 \uppercase{\gdef\macro@#1#2#3#4\macro@{\gtest@false
315 \ifx 5#1\ifx 6#2\ifx 7#3\gtest@true
316 \edef\meaning@{\macro@#4\macro@}\fi\fi\fi}}
317 \def\macro@@#1->#2\macro@@{#2}
318 \newcount\DOTSCASE@
319 {\uccode`6=`\ \uccode`7=`D \uccode`8=`O \uccode`9=`T \uccode`0=`S
320 \uppercase{\gdef\DOTS@#1#2#3#4#5{\gtest@false\DN@##1\DOTS@{}}%
321 \ifx 6#1\ifx 7#2\ifx 8#3\ifx 9#4\ifx 0#5\let\next@DOTS@@
322 \fi\fi\fi\fi\fi
323 \next@}}
324 {\uccode`3=`B \uccode`4=`I \uccode`5=`X
325 \uppercase{\gdef\DOTS@@#1{\relaxnext@
326 \def\nextii@##1\DOTS@{\ifx\@sptoken\@let@token\gtest@true\fi}%
327 \DN@{\FN@\nextii@}%
328 \ifx 3#1\global\DOTSCASE@\z@\else
329 \ifx 4#1\global\DOTSCASE@\one@\else
330 \ifx 5#1\global\DOTSCASE@\tw@\else\DN@##1\DOTS@{}}%
331 \fi\fi\fi\next@}}
332 {\uccode`5=`\ \uccode`6=`n \uccode`7=`o \uccode`8=`t
333 \uppercase{\gdef\not@#1#2#3#4{\relaxnext@
334 \def\nextii@##1\not@{\ifx\@sptoken\@let@token\gtest@true\fi}%
335 \gtest@false\DN@##1\not@{}}%
336 \ifx 5#1\ifx 6#2\ifx 7#3\ifx 8#4\DN@{\FN@\nextii@}\fi\fi\fi
337 \fi\next@}}
338 \def\keybin@{\gtest@true
339 \ifx\@let@token+\else\ifx\@let@token=\else
340 \ifx\@let@token<\else\ifx\@let@token>\else
341 \ifx\@let@token-\else\ifx\@let@token*\else\ifx\@let@token:\else
342 \gtest@false\fi\fi\fi\fi\fi\fi}

```

Patch to ensure \@ldots is defined. (Name changed to \mathellipsis in Dec 94 release of L^AT_EX.)

```

343 \@ifundefined{@ldots}{\def\@ldots{\mathellipsis}}{}

```

Reiterate the standard definition of `\ldots` to keep it from being clobbered by the redefinition of `\dots`.

```

344 \DeclareRobustCommand{\ldots}{%
345   \ifmmode \mathellipsis \else \textellipsis \fi
346 }
347 \DeclareRobustCommand{\dots}{%
348   \ifmmode \@xp\mdots@ \else \@xp\textellipsis \fi
349 }
350 \def\tdots@{\leavevmode\unskip\relaxnext@
351   \DN@{\$ \math\ldots\,
352     \ifx\@let@token,\,$\else\ifx\@let@token.\,$\else
353     \ifx\@let@token;\,$\else\ifx\@let@token:\,$\else
354     \ifx\@let@token? \,$\else\ifx\@let@token! \,$\else
355     $ \fi\fi\fi\fi\fi}%
356   \ \FN@\next@}
357 \def\mdots@{\FN@\mdots@@}
358 \def\mdots@@{\gdef\thedots@{\dotso@}%
359   \ifx\@let@token\boldsymbol \gdef\thedots@\boldsymbol{\boldsymboldots@}%
360   \else\ifx,\@let@token \gdef\thedots@{\dotsc}%
361   \else\ifx\not\@let@token \gdef\thedots@{\dotsb@}%
362   \else\keybin@
363   \ifgtest@\gdef\thedots@{\dotsb@}%
364   \else\xdef\meaning@{\meaning\@let@token.....}%
365   \xdef\meaning@@{\meaning@}%
366   \@xp\math@\meaning@\math@
367   \ifgtest@
368   \@xp\mathch@\meaning@\mathch@
369   \ifgtest@\@xp\getmathch@\meaning@\getmathch@\fi
370   \else\@xp\macro@\meaning@@\macro@
371   \ifgtest@
372   \@xp\not@\meaning@\not@\ifgtest@\gdef\thedots@{\dotsb@}%
373   \else\@xp\DOTS@\meaning@\DOTS@
374   \ifgtest@
375   \ifcase\number\DOTSCASE@\gdef\thedots@{\dotsb@}%
376   \or\gdef\thedots@{\dotsi}\else\fi
377   \else\@xp\math@\meaning@\math@
378   \ifgtest@\@xp\mathbin@\meaning@\mathbin@
379   \ifgtest@\gdef\thedots@{\dotsb@}%
380   \else\@xp\mathrel@\meaning@\mathrel@
381   \ifgtest@\gdef\thedots@{\dotsb@}%
382   \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
383   \thedots@}

```

The `=` character is necessary in the two `\let` assignments in `\boldsymboldots@`, because the symbol we are making bold might be an `=` sign.

```

384 \def\boldsymboldots@#1{%
385   \bold@true\let\@let@token=#1\let\delayed@=#1\mdots@@
386   \boldsymbol#1\bold@false}

```

The definition of `\@cdots` is merely the `plain.tex` definition of `\cdots`.

```

387 \ams@def\@cdots{\mathinner{\cdotp\cdotp\cdotp}}
388 \newcommand{\dotsi}{\!\!\@cdots}
389 \let\dotsb@\@cdots

```

If any new right delimiters are defined, they would need to be added to the definition of `\rightdelim@` in order for `\dots` to work properly in all cases.

```

390 \def\rightdelim@{\gtest@true
391   \ifx\@let@token\else
392   \ifx\@let@token] \else
393   \ifx\@let@token\rbrack \else

```



```

450 \DOTSB\leftarrow\protect\joinrel\protect\relbar}
451 \renewcommand{\Longleftarrow}{%
452 \DOTSB\Leftarrow\protect\joinrel\protect\Relbar}
453 \renewcommand{\longlefttrightarrow}{\DOTSB\leftarrow\joinrel\rightarrow}
454 \renewcommand{\Longlefttrightarrow}{\DOTSB\Leftarrow\joinrel\Rightarrow}
455 \renewcommand{\mapsto}{\DOTSB\mapstochar\rightarrow}
456 \renewcommand{\longmapsto}{\DOTSB\mapstochar\longrightarrow}
457 \renewcommand{\hookrightarrow}{\DOTSB\lhook\joinrel\rightarrow}
458 \renewcommand{\hookleftarrow}{\DOTSB\leftarrow\joinrel\rhook}
459 \renewcommand{\iff}{\DOTSB\; \Longlefttrightarrow\;}

```

The `\doteq` command formerly used `\buildrel`; we avoid that because it requires ‘`\over`’ as part of its syntax. Use `\opt` instead of `\z@` for robustitude.

```

460 \renewcommand{\doteq}{%
461 \DOTSB\mathrel{\mathop{\kern0pt =}\limits^{\textstyle.}}}

```

8 Integral signs

The straightforward `\ifinner` test to see if the current math context is non-display, fails if, for instance, we are typesetting a multiline display within an `\halign`, with the pieces going into constructions like

```
$\displaystyle...$
```

So we need a better test to find out if we are ‘in a display’. We therefore create `\if@display`.

```

462 \newif\if@display
463 \everydisplay\@xp{\the\everydisplay \@displaytrue}

```

Default value for integral limits is `\nolimits`, see the definition of the ‘`nointlimits`’ option.

```

464 \renewcommand{\int}{\DOTSI\intop\ilimits@}
465 \renewcommand{\oint}{\DOTSI\ointop\ilimits@}
466 \def\intkern@{\mkern-6mu\mathchoice{\mkern-3mu}{\}{\}}
467 \def\intdots@{\mathchoice{\@cdots}%
468 {\{\cdotp\}\mkern1.5mu{\cdotp}\mkern1.5mu{\cdotp}}%
469 {\{\cdotp\}\mkern1mu{\cdotp}\mkern1mu{\cdotp}}%
470 {\{\cdotp\}\mkern1mu{\cdotp}\mkern1mu{\cdotp}}}
471 %
472 \ams@newcommand{\iint}{\DOTSI\protect\MultiIntegral{2}}
473 \ams@newcommand{\iiint}{\DOTSI\protect\MultiIntegral{3}}
474 \ams@newcommand{\iiiiint}{\DOTSI\protect\MultiIntegral{4}}
475 \newcommand{\idotsint}{\DOTSI\protect\MultiIntegral{0}}

```

If the `\limits` option is applied, use `\mathop` and fudge the left-hand space a bit to make the subscript visually centered.

`#1` is the multiplicity.

```

476 \newcommand{\MultiIntegral}[1]{%
477 \edef\ints@c{\noexpand\intop
478 \ifnum#1=\z@\noexpand\intdots@\else\noexpand\intkern@\fi
479 \ifnum#1>\tw@\noexpand\intop\noexpand\intkern@\fi
480 \ifnum#1>\thr@\noexpand\intop\noexpand\intkern@\fi
481 \noexpand\intop
482 \noexpand\ilimits@
483 }%
484 \futurelet\@let@token\ints@a
485 }

486 \def\ints@a{%
487 \ifx\limits\@let@token \ints@b

```

```

488 \else \ifx\displaylimits\@let@token \ints@b
489 \else\ifx\ilimits@\displaylimits \ints@b
490 \fi\fi\fi
491 \ints@c
492 }

493 \def\ints@b{%
494 \mkern-7mu\mathchoice{\mkern-2mu}{-}{-}{-}%
495 \mathop\bgroup
496 \mkern7mu\mathchoice{\mkern2mu}{-}{-}{-}%
497 \let\ilimits@\egroup
498 }%

```

9 Size dependent definitions

We now define all stuff which has to change whenever a new math size is to be activated. \LaTeX provides a hook called `\every@math@size` to support such a need. All assignments in the `\every@math@size` hook that need to take outside effect should be global.

9.1 Struts for math

The various kinds of struts could use some analysis and perhaps consolidation.

For example perhaps the `\bBigg` delimiters could use

```
1.2\ht\strutbox (1.8, 2.4, 3.0)
```

instead of

```
1.0\big@size (1.5, 2.0, 2.5)
```

since `\strut` is reset with every size change [mjd, 1994/10/07]. But this change would introduce the possibility of changed line and page breaks in existing documents, so would need to be handled with care.

```

\Mathstrut@ Here comes the code for Spivak's \Mathstrut@.
\Mathstrutbox@ 499 \newbox\Mathstrutbox@
\resetMathstrut@ 500 \setbox\Mathstrutbox@=\hbox{
501 \def\Mathstrut@{\copy\Mathstrutbox@}

```

The setting of the height and depth of the `\Mathstrutbox@` is done in the `\every@math@size` hook since it depends on the height of a paren. As `\every@math@size` is triggered by \$ after a font size change, we want to avoid using another math formula $\$ \dots \$$ to measure the math paren height; instead we go through the mathcode of the (character. We assume that the mathcode has a leading hex digit 4 indicating ‘open delimiter’; this allows us to make a relatively simple function to get the correct font and character position.

```

502 \def\resetMathstrut@{%
503 \setbox\z@\hbox{%
504 \mathchardef\@tempa\mathcode`\(\relax
505 \def\@tempb##1"###3{\the\textfont"###3\char"}%
506 \expandafter\@tempb\meaning\@tempa \relax
507 }%

```

These height and depth assignments are implicitly global.

```

508 \ht\Mathstrutbox@\ht\z@ \dp\Mathstrutbox@\dp\z@
509 }
510 \addto@hook\every@math@size{\resetMathstrut@}

```

```

\strut@ Next follows a special internal strut which is supposed to match the height and
\strutbox@ the depth of a normal \strut minus \normallineskiplimit according to M.
Spivak.

```

This should really go into the definition of `\size@update`, and then the box reset could be local; but `\size@update` doesn't have any hook and is handled in such a way that it cannot even be changed except by changing `\set@fontsize`. So instead we put `\reset@strutbox@` into `\every@math@size` and make it global. Then because of some complications in the way `\glb@settings` and `\check@mathfonts` work, we have to re-invoke it at the beginning of every environment that might use `\strut@`. Fortunately this can be achieved (more or less) through the `\spread@equation` hook. [mjd,2000/03/10]

```

511 \newbox\strutbox@
512 \def\strut@{\copy\strutbox@}
513 \def\reset@strutbox@{%
514   \global\setbox\strutbox@\hbox{%
515     \lower.5\normallineskiplimit
516     \vbox{\kern-\normallineskiplimit\copy\strutbox@}}
517 \addto@hook\every@math@size{\reset@strutbox@}
518 \AtBeginDocument{\reset@strutbox@}

```

9.2 Big delimiters

We are now going to redefine the plain T_EX commands `\big`, `\bigl`, etc., to produce different results in different sizes. Actually we only have to define `\big`, `\Big`, etc., since they are used to construct the directional versions `\bigl`, `\bigr`, and the rest.

`\big` To save token space we put everything into the common macro `\bBigg@`. The
`\Big` macros are now simply a call to `\bBigg@` with a factor to determine the correct
`\bigg` height of the delimiter as an argument. This code should better go into a future
`\Bigg` version of the L^AT_EX kernel; the macro `\n@space` is then superfluous (since it is only used once) and should be removed to avoid wasting hash table space unnecessarily.

```

519 \renewcommand{\big}{\bBigg@\@ne}
520 \renewcommand{\Big}{\bBigg@{1.5}}
521 \renewcommand{\bigg}{\bBigg@{tw@}}
522 \renewcommand{\Bigg}{\bBigg@{2.5}}

```

`\bBigg@` Now we tackle the macro which has to do the real work. It actually has two arguments, the factor and the wanted delimiter.

```

523 \def\bBigg@#1#2{%

```

We start with an extra set of braces because we want constructions like `\def\bigl{\mathopen\big}` to work without the overhead of extra arguments.

```

524   {\@mathmeasure\z@{\nulldelimiterspace\z@}%
525    {\left#2\center to#1\big@size{\right.}%
526    \box\z@}}

```

`\big@size` `\big@size` needs to be set to 1.2 times the height of a math paren. This height is already recorded in `\Mathstrutbox@`.

```

527 \addto@hook\every@math@size{%
528   \global\big@size 1.2\ht\Mathstrutbox@
529   \global\advance\big@size 1.2\dp\Mathstrutbox@ }
530 \newdimen\big@size

```

10 Math accents

We want to change the leading digit of math accents to be `\accentclass@` so that it can vary according to certain internal purposes.

```

531 \def\accentclass@{7}
532 \def\noaccents@{\def\accentclass@{0}}

```

There are a few *math alphabet*s in the standard fonts where we have to change the extra macros because the standard definitions don't account for these accent problems. The first is for the `\mathit` command.

```
533 \DeclareFontEncoding{OML}{\noaccents@}
```

The next one corrects the `\cal` alphabet.

```
534 \DeclareFontEncoding{OMS}{\noaccents@}
```

Triple and quadruple dot accents.

```
535 \ams@newcommand{\dddot}[1]{%
536   {\mathop{#1}\limits^{\vbox to-1.4\ex@{\kern-\tw@ex@
537     \hbox{\normalfont ...}\vss}}}}
538 \ams@newcommand{\ddddot}[1]{%
539   {\mathop{#1}\limits^{\vbox to-1.4\ex@{\kern-\tw@ex@
540     \hbox{\normalfont ...}\vss}}}}
```

The following code deals with support for compound accents. By redefining `\set@mathaccent` we ensure that `\DeclareMathAccent` will define accent commands to run our `\mathaccentV` function instead of the primitive `\mathaccent`.

```
541 \def\set@mathaccent#1#2#3#4{%
542   \xdef#2{\@nx\protect\@nx\mathaccentV
543     {\@xp\@gobble\string#2}\hexnumber@#1#4}%
544 }
```

We redefine the standard math accent commands to call `\mathaccentV`, using the `mathgroup/encoding-number` information embedded in their previous definitions. If the definition of an accent command does not have the expected form, we leave the accent command alone, but give a warning. For widehat and widetilde, we need to avoid clobbering the definitions done by the `amsfonts` package. Arbitrating the contention between `amsmath` and `amsfonts` to allow doubling a widetilde accent looks tricky, so for the time being [mjd,1999/07/19] we just leave `\widehat` and `\widetilde` alone. As a result, if the `amsmath` package is loaded on top of a vanilla L^AT_EX documentclass, everything runs through with no warnings. If a Lucida Math or other math fonts package is loaded in addition to `amsmath`, there are greater difficulties, but those are addressed elsewhere.

```
545 \def\@tempa#1{\@xp\@tempb\meaning#1\@nil#1}
546 \def\@tempb#1>#2#3 #4\@nil#5{%
547   \@xp\ifx\csname#3\endcsname\mathaccent
548     \@tempc#4?"7777\@nil#5%
549   \else
550     \PackageWarningNoLine{amsmath}{%
551       Unable to redefine math accent \string#5}%
552   \fi
553 }
554 \def\@tempc#1"#2#3#4#5#6\@nil#7{%
555   \chardef\@tempd="#3\relax\set@mathaccent\@tempd{#7}{#2}{#4#5}}
556 \@tempa{\hat}
557 \@tempa{\check}
558 \@tempa{\tilde}
559 \@tempa{\acute}
560 \@tempa{\grave}
561 \@tempa{\dot}
562 \@tempa{\ddot}
563 \@tempa{\breve}
564 \@tempa{\bar}
565 \@tempa{\vec}
566 \@ifundefined{mathring}{%
567   \DeclareMathAccent{\mathring}{\mathalpha}{operators}{"17}
```



```

568 }{%
569   \@tempa{\mathring}
570 }
571 %%\@tempa\widetilde
572 %%\@tempa\widehat

```

Regression testing of amsmath 2.0 showed that in some documents there occurred fragments of the form

```
\hat\mathcal{G}
```

This is not at all correct syntax for the argument of a \LaTeX command but it produced the intended result anyway because of the internal syntax of the `\mathaccent` primitive. With `\mathaccentV`, it will yield an error message. We therefore do a special check for such syntax problems in order to make the error message more informative. (dmj: ??????)

```

573 \newcommand{\acc@check}{}
574 \newcommand{\acc@error}{}
575 \def\acc@check{\@ifnextchar\@empty\relax\acc@error}

```

We put most of the tokens in a separate macro so they do not get scanned unless they are actually needed.

```

576 \def\acc@error{%
577   \@amsmath@err{%
578     Improper argument for math accent:\MessageBreak
579     Extra braces must be added to prevent wrong output%
580   }\@ehc
581 }

```

For `\mathaccentV` part of the processing is dependent on the depth of nesting of math accent commands. We introduce a dedicated counter for this instead of using `\chardef` because we want to increment/decrement it during processing, and incrementing a `\chardef` integer is more work.

```
582 \newcount\macc@depth
```

Provide this function in case it is not already available.

```
583 \long\def\@gobblethree#1#2#3{}
```

The `\mathaccentV` function first counts the number of nested math accents by setting the argument in a throw-away box. (This is not as risky as such an operation would normally be because the argument is generally either a simple math symbol or a nested math accent call with a simple math symbol at the bottom of the nesting.)

There are two benefits from counting the nesting levels first before doing anything else: (1) we can fall back to a simple `\mathaccent` call if the nesting depth is 1, and (2) if the nesting depth is greater than 1, we would like to be able to tell when we have reached the lowest level, because at that point we want to save the argument for later use and place an accent on top of a phantom copy.

When we have multiple accents, they will be placed on top of the invisible box, followed by some suitable kerns, then a visible copy of the nucleus. To see why, let us look at what goes wrong with a double application of the `\mathaccent` primitive. The standard definition of `\hat` is `\mathaccent"705E`, so `\hat{\hat{F}}` expands to

```
\mathaccent"705E{\mathaccent"705E{F}}
```

The result of this operation is

```

\ vbox(12.11111+0.0)x7.81946
.\ hbox(6.94444+0.0)x0.0, shifted 1.40973
..\ OT1/cmr/m/n/10 ^
.\ kern-4.30554

```

```

.\vbox(9.47221+0.0)x7.81946
..\hbox(6.94444+0.0)x0.0, shifted 2.24309
...\OT1/cmr/m/n/10 ^
..\kern-4.30554
..\hbox(6.83331+0.0)x7.81946
...\OML/cmm/m/it/10 F

```

TeX starts by constructing a vbox with the hat character on top of the F. Then it puts another hat character on top of the vbox; but without skew information, because that is only applied by `\mathaccent` when the base object is a simple symbol. So the first accent is skewed to the correct position but all later accents are not. By the way, the actual width of the F in the above example is less than 7.81946; the box in which it is packed was automatically lengthened by the width of the F's italic correction (without actually putting in a kern for it).

To get the second accent shifted farther to the right we artificially increase the width of the innermost box and add a compensating kern afterward. Furthermore, to get proper placement of a following subscript or superscript, we take the base symbol out, leaving a phantom in its place, and print it by itself following the kern. We then need to increase the kern amount to move the base character backward under the accents again. Here is what the results look like:

```

\vbox(12.11111+0.0)x9.48618
.\hbox(6.94444+0.0)x0.0, shifted 2.24309
...\OT1/cmr/m/n/10 ^
..\kern-4.30554
.\vbox(9.47221+0.0)x9.48618
..\hbox(6.94444+0.0)x0.0, shifted 2.24309
...\OT1/cmr/m/n/10 ^
..\kern-4.30554
..\hbox(6.83331+0.0)x9.48618
...\hbox(6.83331+0.0)x7.81946
...\kern 1.66672
\kern -9.48618
\OML/cmm/m/it/10 F

```

Much of this implementation is based on code from the `accents` package of Javier Bezos. I added the test to revert to a simple `\mathaccent` when accents are not nested, and some other refinements to reduce the number of kerns used (to conserve box memory) and the number of cycles through `\mathchoice` (to make things run a little faster). It was all rather difficult and my first two attempts had serious bugs but I hope and believe that this version will do better. [mjd,2000/03/15]

The “V” in `\mathaccentV` is just an indication that it takes five arguments. It is important that the name includes `mathaccent`, otherwise `\DeclareMathAccent` will balk at redefining one of our accent commands, for example when an alternative math font package is loaded.

```

584 \def\mathaccentV#1#2#3#4#5{%
585   \ifmmode
586     \gdef\macc@tmp{\macc@depth\@ne}%
587     \setbox\z@\hbox{%
588       \let\mathaccentV\macc@test
589       \let\use@mathgroup\@gobbletwo \let\select@group\@gobblethree
590       \frozen@everymath{}\$#5$%
591     }%
592     \macc@tmp
593     \ifnum\macc@depth=\@ne
594       \global\let\macc@nucleus\@empty
595       \mathaccent"\accentclass@
596     \else

```

```

597     \xp\macc@nested
598     \fi
599     #2#3#4{#5}%
600     \macc@nucleus
601     \else
602     \xp\nonmatherr@{csname#1\endcsname
603     \fi
604 }

605 \def\macc@test#1#2#3#4{\xdef\macc@tmp{\macc@tmp\advance\macc@depth\@ne}}
606 \def\macc@group{-1}
607 \def\macc@nested#1#2#3#4{%
608     \begingroup
609     \let\math@bgroup\@empty \let\math@egroup\macc@set@skewchar
610     \mathsurround\z@ \frozen@everymath{\mathgroup\macc@group\relax}%
611     \macc@set@skewchar\relax
612     \let\mathaccentV\macc@nested@a
613     \macc@nested@a\relax#1#2#3{#4}%
614     \endgroup
615 }

616 \let\macc@palette\mathpalette
617 \def\macc@nested@a#1#2#3#4#5{%

```

This test saves some work that would otherwise be always repeated fourfold thanks to `\mathchoice`.

```

618     \ifnum\macc@group=\mathgroup
619     \else \macc@set@skewchar\relax \edef\macc@group{\the\mathgroup}%
620     \fi
621     \mathchardef\macc@code "\accentclass@ #2#3#4\relax
622     \macc@palette\macc@a{#5}%
623 }

```

The reason that `\macc@set@skewchar` takes an argument is so that it can serve as a direct substitute for `\math@egroup`, in addition to being used separately.

Setting a skewchar with this method works for symbols of variable mathgroup (class 7, letters and numbers) but not necessarily for special symbols like `\partial` or `\xi` whose mathgroup doesn't change; fortunately the most commonly used ones come from mathgroup one, which is the fall-back mathgroup for skewchar.

```

624 \def\macc@set@skewchar#1{%
625     \begingroup
626     \ifnum\mathgroup=\m@ne \let\@tempa\@ne
627     \else
628         \ifnum\skewchar\textfont\mathgroup=\m@ne \let\@tempa\@ne
629         \else \let\@tempa\mathgroup
630     \fi
631     \fi
632     \count@=\skewchar\textfont\@tempa
633     \advance\count@"7100
634     \edef\@tempa{\endgroup
635         \mathchardef\noexpand\macc@skewchar=\number\count@\relax}%
636     \@tempa
637     #1%
638 }

```

Arg1 is math-style, arg2 is accent base object. We assume that math style doesn't change within the nested group of accents; this means we can set `\macc@style` only once and redefine `\macc@palette` to use it, in order to run

`\mathchoice` only once instead of multiplying the calls exponentially as the nesting level increases.

```
639 \def\macc@a#1#2{%
640   \begingroup
641   \let\macc@style#1\relax
642   \def\macc@palette##1{##1\macc@style}%
643   \advance\macc@depth\m@ne
644   \ifnum\macc@depth=\z@
645     \gdef\macc@nucleus{#2}%
```

Extra `\@empty` tokens are to prevent low-level TeX errors from the potential syntactic error that `\acc@check` checks for.

```
646   \setbox\z@\hbox{##1#2\@empty}\macc@skewchar$}%
647   \setbox\tw@\hbox{##1#2\@empty\macc@skewchar$}%
648   \dimen@tw@\wd\tw@ \advance\dimen@-\tw@\wd\z@
649   \xdef\macc@kerna{\the\dimen@\relax}%
650   \setbox4\hbox{##1#2\acc@check\@empty$}%
651   \global\setbox\@ne\hbox to\wd4{}%
652   \ht\@ne\ht4 \dp\@ne\dp4
653   \xdef\macc@kernb{\the\wd4\relax}%
654   \mathaccent\macc@code{\box\@ne\kern\macc@kerna}%
655   \else
656     \mathaccent\macc@code{\let\macc@adjust\@empty #1#2\@empty}%
657     \macc@adjust
658   \fi
659 \endgroup
660 }

661 \def\macc@adjust{%
662   \dimen@\macc@kerna\advance\dimen@\macc@kernb
663   \kern-\dimen@
664 }
```

The commands `\Hat`, `\Tilde`, ..., are supported as synonyms of `\hat`, `\tilde`, ..., for backward compatibility.

```
665 \def\Hat{\hat}
666 \def\Check{\check}
667 \def\Tilde{\tilde}
668 \def\Acute{\acute}
669 \def\Grave{\grave}
670 \def\Dot{\dot}
671 \def\Ddot{\ddot}
672 \def\Breve{\breve}
673 \def\Bar{\bar}
674 \def\Vec{\vec}
```

This error message about math mode is used several times so we make an abbreviation for it.

```
675 \def\nonmatherr#1{\@amsmath@err{\protect
676   #1 allowed only in math mode}\@ehd}
```

11 Mods, continued fractions, etc.

The commands `\bmod`, `\pmod`, `\pod`, `\mod` aren't currently robust. [mjd, 1994/09/05]

```
677 \renewcommand{\bmod}{\nonscript\mskip-\medmuskip\mkern5mu\mathbin
678   {\operator@font mod}\penalty900
679   \mkern5mu\nonscript\mskip-\medmuskip}
680 \newcommand{\pod}[1]{\allowbreak
681   \if@display\mkern18mu\else\mkern8mu\fi(#1)}
```

```

682 \renewcommand{\pmod}[1]{\pod{\operator@font mod}\mkern6mu#1}}
683 \newcommand{\mod}[1]{\allowbreak\if@display\mkern18mu
684   \else\mkern12mu\fi{\operator@font mod}\,\,\#1}

```

Continued fractions. The optional arg l or r controls horizontal placement of the numerators. The `\kern-\nulldelimiterspace` is needed in the definition if we want the right-hand sides of the fraction rules to line up. The `\strut` keeps the numerator of a subsidiary cfrac from coming too close to the fraction rule above it.

```

685 \newcommand{\cfrac}[3][c]{\displaystyle\frac{%
686   \strut\ifx r#1\hfill\fi#2\ifx l#1\hfill\fi}{#3}}%
687 \kern-\nulldelimiterspace}

```

`\overset` and `\underset` put symbols above, respectively below, a symbol that is not a `\mathop` and therefore does not naturally accept limits. `\binrel@` uses information collected by `\binrel@` to make the resulting construction be of type `mathrel` or `mathbin` if the base symbol is either of those types.

```

688 \newcommand{\overset}[2]{\binrel@{#2}%
689   \binrel@{\mathop{\kern\z@#2}\limits^{#1}}}
690 \newcommand{\underset}[2]{\binrel@{#2}%
691   \binrel@{\mathop{\kern\z@#2}\limits_{#1}}}

```

`\sideset` allows placing ‘adscript’ symbols at the four corners of a `\mathop`, *in addition to* limits. Left-side adscripts go into arg #1, in the form `_{\dots}^{\dots}`, and right-side adscripts go into arg #2.

As currently written [mjd, 1995/01/21] this is pretty haphazard. In order to really make it work properly in full generality we’d have to read and measure the top and bottom limits and use `mathchoice` to always get the right `mathstyle` for each piece, etc., etc.

```

692 \newcommand{\sideset}[3]{%
693   \@mathmeasure\z@\displaystyle{#3}%

```

Use a global box assignment here since the depth override is implicitly global. Then move the constructed box to a local box register (2) to ensure it won’t get destroyed during the next two `mathmeasure` statements. This precaution may be more extreme than necessary in practice.

```

694   \global\setbox\@ne\vbox to\ht\z@{\dp\@ne\dp\z@
695   \setbox\tw@box\@ne
696   \@mathmeasure4\displaystyle{\copy\tw@#1}%
697   \@mathmeasure6\displaystyle{#3\nolimits#2}%
698   \dimen@-\wd6 \advance\dimen@\wd4 \advance\dimen@\wd\z@
699   \hbox to\dimen@{\mathop{\kern-\dimen@box4box6}%
700 }

```

`\smash` We add to the `\smash` command an optional argument denoting the part of the formula to be smashed.

```

701 \renewcommand{\smash}[1][tb]{%
702   \def\mb@t{\ht}\def\mb@b{\dp}\def\mb@tb{\ht\z@\z@\dp}%
703   \edef\finsh{\csname mb@#1\endcsname\z@\z@ \box\z@}%
704   \ifmode \xp\mathpalette\exp\mathsm@sh
705   \else \xp\makesm@sh
706   \fi
707 }

```

12 Extensible arrows

The minus sign used in constructing these arrow fills is smashed so that superscripts above the arrows won’t be too high. This primarily affects the `\xleftarrow` and `\xrightarrow` arrows.

```

708 \mathchardef\std@minus\mathcode`\-\relax
709 \mathchardef\std@equal\mathcode`\=\relax

```

In case some alternative math fonts are loaded later:

```

710 \AtBeginDocument{%
711   \mathchardef\std@minus\mathcode`\-\relax
712   \mathchardef\std@equal\mathcode`\=\relax
713 }

714 \ams@def\relbar{\mathrel{\mathpalette\mathsm@sh\std@minus}}
715 \ams@def\Relbar{\mathrel{\std@equal}}

716 \def\arrowfill@#1#2#3#4{%
717   $\m@th\thickmuskip0mu\medmuskip\thickmuskip\thinmuskip\thickmuskip
718   \relax#4#1\mkern-7mu%
719   \cleaders\hbox{$#4\mkern-2mu#2\mkern-2mu$}\hfill
720   \mkern-7mu#3$%
721 }

722 \def\leftarrowfill@{\arrowfill@\leftarrow\relbar\relbar}
723 \def\rightarrowfill@{\arrowfill@\rightarrow\relbar\relbar}
724 \def\leftrightarrowfill@{\arrowfill@\leftarrow\relbar\rightarrow}
725 \def\Leftarrowfill@{\arrowfill@\Leftarrow\Relbar\Relbar}
726 \def\Rightarrowfill@{\arrowfill@\Rightarrow\Relbar\Relbar}
727 \def\Leftrightarrowfill@{\arrowfill@\Leftrightarrow\Relbar\Relbar}

728 \def\overarrow@#1#2#3{\vbox{\ialign{##\crrc#1#2\crrc
729   \noalign{\nointerlineskip}}$\m@th\hfil#2#3\hfil$\crrc}}
730 \ams@renewcommand{\overrightarrow}{%
731   \mathpalette{\overarrow@\rightarrowfill@}}
732 \ams@renewcommand{\overleftarrow}{%
733   \mathpalette{\overarrow@\leftarrowfill@}}
734 \ams@newcommand{\overleftrightarrow}{%
735   \mathpalette{\overarrow@\leftrightarrowfill@}}

736 \def\underarrow@#1#2#3{%
737   \vtop{\ialign{##\crrc$\m@th\hfil#2#3\hfil$\crrc
738   \noalign{\nointerlineskip\kern1.3\ex@}\#1#2\crrc}}
739 \ams@newcommand{\underrightarrow}{%
740   \mathpalette{\underarrow@\rightarrowfill@}}
741 \ams@newcommand{\underleftarrow}{%
742   \mathpalette{\underarrow@\leftarrowfill@}}
743 \ams@newcommand{\underleftrightarrow}{%
744   \mathpalette{\underarrow@\leftrightarrowfill@}}

745 %\newcommand{\xrightarrow}[2][\ext@arrow 0359\rightarrowfill@{\#1}{\#2}]
746 \def\ext@arrow#1#2#3#4#5#6#7{%
747   \mathrel{\mathop{

```

Measure the superscript and subscript.

```

748   \setbox\z@\hbox{\#5\displaystyle}%
749   \setbox\tw@\vbox{\m@th
750     \hbox{$\scriptstyle\mkern#3mu{\#6}\mkern#4mu$}%
751     \hbox{$\scriptstyle\mkern#3mu{\#7}\mkern#4mu$}%
752     \copy\z@
753   }%
754   \hbox to\wd\tw@{\unhbox\z@}}%

```

We don't want to place an empty subscript since that will produce too much blank space below the arrow.

```

755   \limits
756   \@ifnotempty{\#7}{~{\if0#1\else\mkern#1mu\fi
757     #7\if0#2\else\mkern#2mu\fi}}%
758   \@ifnotempty{\#6}{~{\if0#1\else\mkern#1mu\fi

```

```

759             #6\if0#2\else\mkern#2mu\fi}}}%
760 }

```

Some extensible arrows to serve as mathrels and taking sub/superscripts. These commands are robust because they take an optional argument.

```

761 \newcommand{\xrightarrow}[2][]{\ext@arrow 0359\rightarrowfill@{#1}{#2}}
762 \newcommand{\xleftarrow}[2][]{\ext@arrow 3095\leftarrowfill@{#1}{#2}}

```

13 Array-related environments

13.1 Remarks

Because these environments can be nested within the equation structures that allow `\tag`, there is some cross-influence in the internal workings of the `\` command.

13.2 The `\substack` command

The `\substack` command can be used to set subscripts and superscripts that consist of several lines. Usage:

```
X_{\substack{a=1\\b=2}}
```

subarray The `subarray` environment makes a small-size array suitable for use in a subscript or superscript. At the moment the supported arguments are not the full possibilities of `array` but only `c` or `l` for centered or left-aligned. And only one column.

```
763 \newenvironment{subarray}[1]{%
```

Note: The predecessors of `subarray` (`Sb` and `Sp`, inherited from $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\mathcal{T}\mathcal{E}\mathcal{X}$) used `\vbox` instead of `\vcenter`. But when a multiline subscript is placed in `\limits` position `\vcenter` is no worse than `\vbox`, and when it is placed in the `\nolimits` position (e.g., for an integral), `\vcenter` provides clearly better positioning than `\vbox`.

```
764 \vcenter\bgroup
```

Use `\Let@` to set the proper meaning of the `\` and `*` commands. And restore the meaning of `\math@cr@@@` to `\cr` (see above) in case `subarray` is used inside one of the more complicated alignment macros where the meaning of `\math@cr@@@` is different. Similarly, call `\default@tag` to ensure that a line break here doesn't get an equation number!

```
765 \Let@ \restore@math@cr \default@tag
```

Set the line spacing to be the same as `\atop` (when `\atop` occurs in `\textstyle` or smaller), cf *The T_EXbook*, Appendix G.

```

766 \baselineskip\fontdimen10 \scriptfont\tw@
767 \advance\baselineskip\fontdimen12 \scriptfont\tw@
768 \lineskip\thr@@\fontdimen8 \scriptfont\thr@@
769 \lineskiplimit\lineskip

```

Start the `\vbox \halign` structure that encloses the contents. Notice that we never get `\scriptscriptstyle`. That would require a `\mathchoice` (ugh).

```

770 \ialign\bgroup\ifx c#1\hfil\fi
771     $\m@th\scriptstyle##$\hfil\cr
772 }{%
773 \cr\egroup\egroup
774 }

```

\substack The `\substack` command is just an abbreviation for the most common use of `subarray`.

```
775 \newcommand{\substack}[1]{\subarray{c}#1\endsubarray}
```

13.3 Matrices

smallmatrix `smallmatrix` is again an alignment, this time in a centered box. The opening incantations are basically the same as those in `\multilimits@`, followed by the alignment itself. A remark: the `baselineskip` (9\ex@) used in $\mathcal{A}\mathcal{M}\mathcal{S}$ - \TeX is too large for use in text with the usual `baselineskip` of 12 or 13 points; we change it here to 6\ex@ and also adjust the `\lineskip` and `\lineskiplimit` slightly to compensate. (MJD)

```

776 \newenvironment{smallmatrix}{\null\,\vcenter\bgroup
777   \let\restore@math@cr\default@tag
778   \baselineskip6\ex@ \lineskip1.5\ex@ \lineskiplimit\lineskip
779   \ialign\bgroup\hfil$\m@th\scriptstyle##$\hfil&&\thickspace\hfil
780   $\m@th\scriptstyle##$\hfil\cr
781 }{%
782   \cr\egroup\egroup\,%
783 }
```

matrix The `matrix` environment is just an `array` that provides up to ten centered columns, so that users don't have to give the `col-spec` argument explicitly—unless they want some of the columns noncentered, that is. The maximum number of columns is actually not fixed at ten but given by the counter `MatrixCols`, and can therefore be increased by changing that counter.

The extra space of `\arraycolsep` that `array` adds on each side is a waste so we remove it here (perhaps we should instead remove it from `array` in general, but that's a harder task).

TODO: Think about re-implementing `\matrix` to get rid of the `\c@MatrixCols` limit and have hard-wired preamble that doesn't have to be rebuilt each time.

We must use `\renewenvironment` for `matrix` and `pmatrix` because \LaTeX doesn't kill the definitions found in `plain.tex`, even though it probably should because of their foreign syntax.

```

784 \renewenvironment{matrix}{%
785   \matrix@check\matrix\env@matrix
786 }{%
787   \endarray \hskip -\arraycolsep
788 }
```

`\env@matrix`

```

789 \def\env@matrix{\hskip -\arraycolsep
790   \let\@ifnextchar\new@ifnextchar
791   \array{*\c@MaxMatrixCols c}}
```

`\c@MaxMatrixCols`

```

792 \newcount\c@MaxMatrixCols \c@MaxMatrixCols=10
```

\matrix@check For various reasons, authors sometimes use the Plain \TeX form of `\matrix` or `\pmatrix` in \LaTeX documents. If they later add an invocation of the `amsmath` package to their document, the Plain \TeX syntax would lead to rather unintelligible error messages. The `\matrix@check` function does some checking to forestall that problem.

```

793 \def\matrix@check#1{%
794   \@xp\ifx\csname\@currenenv\endcsname#1%
795   \else\matrix@error#1%
```

This error recovery is not that good but is better than the infinite loop that can result from calling `\array` without a matching `\endarray`. (The array setup leaves `\par` empty.)

```

796   \@xp\@gobble
797   \fi
```



```

798 }

\matrix@error

799 \def\matrix@error#1{%
800   \@amsmath@err{%
801     Old form ``\string#1' should be \string\begin{\@xp\@gobble\string#1}%
802   }{%
803     ``\string#1{...}' is old Plain-Tex syntax whose use is
804     ill-advised in LaTeX.%
805   }%
806 }

807 \renewenvironment{pmatrix}{%
808   \left(%
809   \matrix@check\pmatrix\env@matrix
810 }{
811   \endmatrix\right)%
812 }
813 \newenvironment{bmatrix}{\left[\env@matrix}{\endmatrix\right]}
814 \newenvironment{Bmatrix}{%
815   \left\lbracket\env@matrix
816 }{%
817   \endmatrix\right\rbrace
818 }
819 \newenvironment{vmatrix}{\left\lvert\env@matrix}{\endmatrix\right\rvert}
820 \newenvironment{Vmatrix}{\left\lVert\env@matrix}{\endmatrix\right\lVert}
821 \let\hdots\@ldots

822 \newcommand{\hdotsfor}[1]{%
823   \ifx[#1\@xp\shdots@for\else\hdots@for\@ne{#1}\fi}
824 \newmuskip\dotsspace@
825 \def\shdots@for#1{\hdots@for{#1}}
826 \def\hdots@for#1#2{\multicolumn{#2}c%
827   {\m@th\dotsspace@1.5mu\mkern-#1\dotsspace@
828     \xleaders\hbox{$\m@th\mkern#1\dotsspace@.\mkern#1\dotsspace@$}%
829     \hfill
830     \mkern-#1\dotsspace@}%
831   }

```

cases The easiest way to produce the `cases` environment is to base it on the `array` environment. We must use `\renewenvironment` to override the definition of `\cases` that L^AT_EX (unwisely) leaves in place from `plain.tex`.

```

832 \renewenvironment{cases}{%
833   \matrix@check\cases\env@cases
834 }{%
835   \endarray\right.%
836 }
837 \def\env@cases{%
838   \let\@ifnextchar\new@ifnextchar
839   \left\lbrace
840   \def\arraystretch{1.2}%
841   \array{@{}l@{\quad}l@{}}%
842 }

```

14 Equation sub-numbering

```

843 \newcounter{parentequation}% Counter for ``parent equation''.

```

We can't assume `\ignorespacesafterend` is defined since it was not there in the earliest releases of L^AT_EX 2e. And we need to include the `\global` for the same reason.

```

844 \ifundefined{ignorespacesafterend}{%
845   \def\ignorespacesafterend{\global\@ignoretrue}%
846 }{}
847 \newenvironment{subequations}{%

```

Before sending down the ‘equation’ counter to the subordinate level, add 1 using standard `\refstepcounter`.

```
848 \refstepcounter{equation}%
```

Define `\theparentequation` equivalent to current `\theequation`. `\edef` is necessary to expand the current value of the equation counter. This might in rare cases cause something to blow up, in which case the user needs to add `\protect`.

```

849 \protected@edef\theparentequation{\theequation}%
850 \setcounter{parentequation}{\value{equation}}%

```

And set the equation counter to 0, so that the normal incrementing processes in the various equation environments will produce the desired results.

```

851 \setcounter{equation}{0}%
852 \def\theequation{\theparentequation\alph{equation}}%
853 \ignorespaces
854 }{%
855 \setcounter{equation}{\value{parentequation}}%
856 \ignorespacesafterend
857 }

```

15 Equation numbering

In the multiline equation environments provided here, the task of equation numbering is linked to the task of line breaking in the sense that it is the `\` command that marks where an equation number for the current line will be processed and added to the page.

Provide a convenient way to specify that equations should be numbered within sections.

```

858 \newcommand{\numberwithin}[3][\arabic]{%
859   \@ifundefined{c#2}{\@nocounterr{#2}}{%
860     \@ifundefined{c#3}{\@nocnterr{#3}}{%
861       \@addtoreset{#2}{#3}%
862       \@xp\xdef\csname the#2\endcsname{%
863         \@xp\@nx\csname the#3\endcsname .\@nx#1{#2}}}%
864 }

```

To make references to equation numbers easier, we provide `\eqref`. We almost don’t need `\textup`, except that `\tagform@` doesn’t supply the italic correction.

```
865 \newcommand{\eqref}[1]{\textup{\tagform@\ref{#1}}}
```

15.1 Preliminary macros

The following macros implement the \LaTeX syntax for the `\` command, i.e. the possibility to add an asterisk to inhibit a page break, or an optional argument to denote additional vertical space. They are modelled more or less after the corresponding macros for \LaTeX ’s `eqnarray` and `array` environments.

[We can perhaps use the `eqnarray` mechanism if we change it so that it also uses `\openup`.]

`\dspbrk@lv1` We begin by defining the `\dspbrk@lv1` counter. This counter records the desirability of a break after the current row, as a number between 0 and 4. Its default value is -1 meaning that no explicit `\displaybreak` command was given, and the default `\interdisplaylinepenalty` is to be used.

```
866 \newcount\dspbrk@lv1
```

```

867 \dspbrk@lvl=-1

rdisplaylinepenalty We set the \interdisplaylinepenalty to 10000.
868 \interdisplaylinepenalty\@M

\allowdisplaybreaks The \allowdisplaybreaks command. Since this is intended for use out-
side displayed formulas (typically in the preamble), it does not need to use
\new@ifnextchar.
869 \newcommand{\allowdisplaybreaks}[1][4]{%
870   \interdisplaylinepenalty\getdsp@pen{#1}\relax
871 }

\getdsp@pen Modelled after LATEX's \@getpen. We use higher numbers than would normally
be provided by \@lowpenalty, \@medpenalty, and \@highpenalty, since dis-
play breaks are almost always less desirable.
872 \def\getdsp@pen#1{%
873   \ifcase #1\@M \or 9999 \or 6999 \or 2999 \or \z@\fi
874 }

\displaybreak For breaks in a certain row of a alignment.
\dspbrk@ 875 \newcommand{\displaybreak}{\new@ifnextchar[\dspbrk@\dspbrk@{4}]}
\dspbrk@context 876 \chardef\dspbrk@context=\sist@on
nogood@displaybreak 877 \def\dspbrk@[#1]{%
878   \ifmeasuring@
879   \else
880     \ifcase\dspbrk@context % case 0 --- OK
881       \global\dspbrk@lvl #1\relax
882     \or % case 1 --- inside a box
883       \nogood@displaybreak
884     \else % other cases --- outside of a display
885       \@amsmath@err{\Invalid@\displaybreak}\@eha
886     \fi
887   \fi
888 }

This is the value of \displaybreak when it occurs inside some structure
where it will not work.
889 \def\nogood@displaybreak{%
890   \@amsmath@err{\protect
891 \displaybreak\space cannot be applied here}%
892 {One of the enclosing environments creates an
893 unbreakable box\MessageBreak
894 (e.g., split, aligned, gathered, ...).}%
895 }

\math@cr The macro \math@cr ends a row inside one of the equation environments, i.e.,
this is the internal name of the \ commands in these environments. As usual
for this kind of macro inside of alignments we insert a special brace into TEX's
input stream. The initial \relax is needed to trigger entry into the u template
of the current column if the author ended the current row with an empty column
(i.e., the mathcr was immediately preceded by an ampersand).
896 \def\math@cr{\relax\iffalse{\fi\ifnum0=}\fi

The first step is now to check whether an asterisk follows. \@eqpen is used to
hold the penalty value to be put on the vertical list. Then we call up \math@cr@
which performs the next step. If an asterisk is read page breaking is inhibited.
897 \@ifstar{\global\@eqpen\@M\math@cr}%

```

Otherwise we have to check the `\dspbrk@lvl` value.

```

898      {\global\@eqpen
899      \ifnum\dspbrk@lvl <\z@ \interdisplaylinepenalty
900      \else -\@getpen\dspbrk@lvl \fi
901      \math@cr@}}

```

`\math@cr@` The purpose of `\math@cr@` is to check whether an optional argument follows. If not it provides `\z@` as default value.

```
902 \def\math@cr@{\new@ifnextchar[\math@cr@@{\math@cr@@[\z@]}}
```

`\math@cr@@` `\math@cr@@` closes the special brace opened in `\math@cr`, and calls `\math@cr@@@` which is supposed the ‘real’ row ending command. The meaning of this macro depends on the environment in which it is used.

```
903 \def\math@cr@@[#1]{\ifnum0=\fi \iffalse}\fi\math@cr@@@
```

Finally we put the additional space onto the vertical list.

```
904 \noalign{\vskip#1\relax}}
```

`\Let@` `\Let@` is called by all environments where `\` ends a row of an alignment.

```
905 \def\Let@{\let\\\math@cr}
```

`\restore@math@cr` We mentioned already that the exact meaning of `\math@cr@@@` depends on the current environment. Since it is often a simple `\cr` we provide `\restore@math@cr` to reset it.

```
906 \def\restore@math@cr{\def\math@cr@@@{\cr}}
```

This is also the default case.

```
907 \restore@math@cr
```

`\intertext` The `\intertext` command is used for inserting text between the rows of an alignment. It might better be done as an environment, but the `\begingroup` from `\begin` would cause the `\noalign` to fail.

```
908 \newcommand{\intertext}{\@amsmath@err{\Invalid@@\intertext}\@eha}
```

`\intertext@` is called by all environments that allow the use of the `\intertext` command.

```
909 \def\intertext@{%
910 \def\intertext##1{%
```

If current mode is not vmode, the most likely reason is that the writer forgot the `\` that is supposed to precede `\intertext`. All right, then, let’s try adding it our ownself. But, to be slightly careful: `\` does a futurelet, and it’s slightly dangerous to allow a letted token to barge around loose in our internal code when it has been let to a conditional token like `\fi`. So let’s interpose something in front of the `\fi` for the futurelet to take instead. (And careful again: it has to be something evanescent, not (e.g.) `\relax` which would cause the next halign cell to fire up and keep `\noalign` from working.)

```

911 \ifvmode\else\\\@empty\fi
912 \noalign{%
913 \penalty\postdisplaypenalty\vskip\belowdisplayskip
914 \vbox{\normalbaselines

```

We need to do something extra if the outside environment is a list environment. I don’t see offhand an elegant way to test “are we inside any list environment” that is both easy and reliable (for example, checking for zero `\@totalleftmargin` wouldn’t catch the case where `\@totalleftmargin` is zero but `\linewidth` is less than `\columnwidth`), so it seems to me checking `\linewidth` is the best practical solution.

```

915      \ifdim\linewidth=\columnwidth
916      \else \parshape\@ne \@totalleftmargin \linewidth
917      \fi
918      \noindent##1\par}%
919      \penalty\predisplaypenalty\vskip\abovedisplayskip%
920    }%
921  }}

```

15.2 Implementing tags and labels

In this section we describe some of the macros needed to make the `\tag` command work in various places. We start by defining a help text to be used when a `\tag` command is used somewhere it should not appear.

`\tag@help` This is the default error help text provided when `\tag` generates an error message. Note that `\newhelp` generates a control sequence name from the string given as its argument so that a leading backslash is provided automatically.

```

922 \newhelp\tag@help
923 {tag cannot be used at this point.\space
924   If you don't understand why^^Jyou should consult
925   the documentation.^^JBut don't worry: just continue, and I'll
926   forget what happened.}

```

`\gobble@tag` This macro is to be used when `\tag` should silently skip its argument. It is made to handle the `*`-form of `\tag` as well.

```

927 \def\gobble@tag{\@ifstar\@gobble\@gobble}

```

`\invalid@tag` `\invalid@tag` is a macro that should be used whenever `\tag` appears in an illegal place. It sets up `\tag@help` (as defined above) as help message, prints its argument as error message, and skips `\tag`'s argument.

```

928 \def\invalid@tag#1{\@amsmath@err{#1}{\the\tag@help}\gobble@tag}

```

`\dft@tag` `\dft@tag` provides a convenient way to disallow the use of `\tag` at certain points. One simply has to write

```
\let\tag\dft@tag
```

and the `\tag` command will produce an error message, with a suitable error help text, and discard its argument.

```

929 \def\dft@tag{\invalid@tag{\string\tag\space not allowed here}}

```

Since this is used several times we provide an abbreviation for it.

```

930 \def\default@tag{\let\tag\dft@tag}

```

Since this is also the default, i.e. the `\tag` command should not be used except in special places, we issue a `\default@tag` command.

```

931 \default@tag

```

Now that we have taken care of the case that `\tag` is not allowed we will provide some macros to process tags appropriately. As the user documentation states, a `\tag` command (without the asterisk) typesets its argument according to the document styles' conventions, whereas a `\tag*` command typesets its argument exactly as given. We define therefore the following interface:

`\maketag@@` `\tag` is supposed to call `\maketag@@` which checks whether an asterisk follows. If this is the case it calls up `\maketag@@@` which sets its argument 'as is'. Otherwise `\tagform@` is called to do the job. (This macro is to be defined appropriately by the document style.)

```

932 \def\maketag@@{\@ifstar\maketag@@@\tagform@}

```

We define `\maketag@@@` to use the normal font of the document text (since this is the usual practice for numbering of document elements) and to put a box around the tag. Furthermore we use `\m@th` for exceptional cases where the tag involves a superscript or some such math. (Probably from an explicit use of `\tag*` rather than from the automatic numbering.)

```
933 \def\maketag@@@#1{\hbox{\m@th\normalfont#1}}
```

We use the following default definition for `\tagform@` that puts only parentheses around the tag.

```
934 \def\tagform@#1{\maketag@@@{(\ignorespaces#1\unskip\@italiccorr)}}
```

We need to insinuate `\tagform@` into `\@eqnnum` in case `eqnarray` is used (probably in a document that was originally written without use of the `amsmath` package).

```
935 \iftagsleft@
936 \def\@eqnnum{\hbox to1sp{\rlap{\normalfont\normalcolor
937 \hskip -\displaywidth\tagform@\theequation}}}
938 \else
939 \def\@eqnnum{{\normalfont\normalcolor \tagform@\theequation}}
940 \fi
```

`\thetag` Sometimes one needs to set a literal tag according to the rules of the document style. To achieve this we provide the `\thetag` command. It typesets its argument by calling `\tagform@` on it.

```
941 \newcommand{\thetag}{\leavevmode\tagform@}
```

`\df@tag` Sometimes it is necessary for a `\tag` command to store a tag in a safe place and to process it later, e.g., for a tag in a row of an alignment where the tag can only be typeset when the `\\` at the end of the row was seen. Such a tag is stored in the macro `\df@tag` (for ‘deferred tag’). For this purpose we provide the `\make@df@tag` macro. It is built very similar to the `\maketag@@@` macro above.

```
942 \let\df@tag\@empty
943 \def\make@df@tag{\ifstar\make@df@tag@@\make@df@tag@@@}
```

`\make@df@tag` sets `\@currentlabel` and defines `\df@tag` appropriately.

To simplify the task of tracking `\tag` and `\label` commands inside math display environments, we defer `\label` commands until the tag is typeset, similar to the way that `\tags` themselves are deferred. This allows arbitrary placement of `\label` and `\tag` commands and also means we only increment the `\equation` counter when we really need to, thus avoiding the `\setb@ck` nonsense that used to be required.

```
944 \def\make@df@tag@@#1{%
945 \gdef\df@tag{\maketag@@@{#1}\def\@currentlabel{#1}}}
```

Autogenerated number:

```
946 \def\make@df@tag@@@#1{\gdef\df@tag{\tagform@{#1}%
947 \toks@{\@xp{\p@equation{#1}}\edef\@currentlabel{\the\toks@}}}
```

`\ltx@label` Next, we store the default definition of `\label` in `\ltx@label` and then define a new version of `\label` for use in math display environments. `\label@in@display` merely issues a warning message if there is already a pending label (which will be discarded) and then stores the label in `\df@label`.

```
948 \let\ltx@label\label
949 %
950 \def\label@in@display{%
951 \ifx\df@label\@empty\else
```

```

952      \amsmath@err{Multiple \string\label's:
953      label '\df@label' will be lost}\@eha
954    \fi
955    \gdef\df@label
956  }

```

In case there is an `enumerate` inside a `minipage` inside an equation, we need to reset `\label` to its normal value:

```

957 \toks@{\xp{\arrayparboxrestore \let\label\ltx@label}}%
958 \edef\@arrayboxrestore{\the\toks@}
959 \let\df@label\@empty

```

`\make@display@tag`

Now we define a macro to process `\tag` and `\label` commands in various display environments. If the `@eqnsw` switch is set, then we should supply an equation number; otherwise, if the `@tag` switch is set, we should use the tag stored in `\df@tag`. Finally, we process any pending `\labels`.

TODO: Arguably, `\make@display@tag` should issue a warning message if there is a `\label` but neither a tag nor an equation number. Also, it would probably be worthwhile to explore whether `\iftag@` could be done away with and replaced by checks to see if `\df@tag` is empty or not.

```

960 \def\make@display@tag{%
961   \if@eqnsw \incr@eqnum \print@eqnum
962   \else \iftag@ \df@tag \global\let\df@tag\@empty \fi
963   \fi

```

Need to check the `\ifmeasuring@` flag otherwise the `\write` node from `\label` might be discarded in a temp box and clearing `\df@label` will keep it from being reiterated on the real typesetting pass.

```

964   \ifmeasuring@
965   \else
966     \ifx\df@label\@empty
967     \else
968       \xp\ltx@label\xp{\df@label}%
969       \global\let\df@label\@empty
970     \fi
971   \fi
972 }

```

Now we define the special versions of `\tag` used within the `align` environments.

`\tag@in@align`

The `\tag` command may only appear once in a row of an alignment. Therefore we first check the switch `tag@` that is set to false at the begin of every row. If this switch is true a `\tag` was already given in this row and we define `\next@` to expand to a call to `\invalid@tag`.

```

973 \def\tag@in@align{%
974   \relax
975   \iftag@
976     \DN@{\invalid@tag{Multiple \string>tag}}%
977   \else

```

Otherwise we set the `tag@` switch. But there is more to be done: we must also prevent the automatic generation of a tag. Therefore we also reset the `@eqnsw`.

```

978   \global\tag@true

```

Changed to `\nonumber`, since that seems to be all that's required.—dmj, 1994/12/21

```

979   \nonumber

```

Within a row of an `align` environment the `\tag` command must not typeset the tag immediately since its position can be determined only later. Therefore we use the `\make@df@tag` macro defined earlier. Finally we call `\next@` to process the argument that follows.

```

980      \let\next@\make@df@tag
981      \fi
982      \next@
983 }

```

`\raisetag` Usage: `\raisetag <dimen>`

This will modify the vertical placement of the tag of the current equation by `<dimen>`. Note that according to the current uses of `\raise@tag` in e.g., `\place@tag@gather`, no adjustment occurs if the tag falls in its normal position; i.e., `\raisetag` has no effect unless the tag has already been shifted off-line.

```

984 \newcommand{\raisetag}[1]{\skip@#1\relax
985   \xdef\raise@tag{\vskip\iftagsleft@else-\fi\the\skip@\relax}%
986 }

```

`\raise@tag` will be reemptied at the beginning of each equation, which might occur at a `\begin{xxx}` or `\.`

```

987 \let\raise@tag\@empty

```

`\notag` For consistency we provide `\notag`, equivalent to `\nonumber`. The alternative would have been to rename `\tag` as `\number` to go along with `\nonumber`, but of course `\number` is a TeX primitive that should not be redefined.

```

988 \newcommand{\notag}{\nonumber}

```

`\nonumber` Need to add some additional code to `\nonumber` to deal with some complications related to nested environments.

```

989 \renewcommand{\nonumber}{%
990   \if@eqnsw
991     \ifx\incr@eqnum\@empty \addtocounter{equation}\m@ne \fi
992   \fi
993   \let\print@eqnum\@empty \let\incr@eqnum\@empty
994   \global\@eqnswfalse
995 }

996 \def\print@eqnum{\tagform@\theequation}
997 \def\incr@eqnum{\refstepcounter{equation}\let\incr@eqnum\@empty}

```

16 Multiline equation environments

16.1 Remarks

In late 1994 David M. Jones did a thorough overhaul of these environments so that the number placement and a few other aspects are substantially improved over the original versions that were ported essentially unchanged from `amstex.tex` in 1989. Most of the commentary in this section is DMJ's, and comments of any significance that I added are marked by my initials and date [mjd, 1995/01/11].

16.2 Preliminaries

`\iffinalign@` We define two switches that are set to true in certain alignments: `inalign@` and `\ifingather@` inside of the `align` and `gather` environments. These switches are needed to control certain actions that depend on the surrounding conditions, more specifically: on the setting already done by the surrounding environments.

```

998 \newif\iffinalign@
999 \newif\ifingather@

```


Historical Note: Removed the `\ifinany@` test [mjd,1999/06/28] since it was mainly used for the purpose now handled by `\spread@equation`.

```
@arrayparboxrestore Here we must reset a few additional parameters.
1000 \exp\def\exp\@arrayparboxrestore\exp{\@arrayparboxrestore
1001   \ingather@false\inalign@false \default@tag
1002   \let\spread@equation\@spread@equation
1003   \let\reset@equation\@empty
1004   \def\print@eqnum{\tagform@\theequation}%
1005   \def\incr@eqnum{\refstepcounter{equation}\let\incr@eqnum\@empty}%
1006 }

\iftag@ The switch tag@ is set to false at the beginning of every row and set to true by
a \tag command. This allows us to check whether there is more than one tag
on a row.
1007 \newif\iftag@

\ifst@rred The switch st@rred is set to true by all starred environments and set to false
by the unstarred versions.
1008 \newif\ifst@rred

\ifmeasuring@ All display environments get typeset twice—once during a “measuring” phase
and then again during a “production” phase; \ifmeasuring@ will be used to
determine which case we’re in, so we can take appropriate action.
1009 \newif\ifmeasuring@

\ifshifttag@ \ifshifttag@ is used by gather to communicate between \calc@shift@gather
and \place@tag@gather whether an equation tag should be shifted to a sepa-
rate line. It’s also used by multline.
1010 \newif\ifshifttag@

\row@
1011 \newcount\row@

\column@ The counter \column@ is used by the alignment macros to keep track of the
current column.
1012 \newcount\column@

\column@plus \column@plus is a useful abbreviation.
1013 \def\column@plus{%
1014   \global\advance\column@\@ne
1015 }

\maxfields@
1016 \newcount\maxfields@

\add@amp
\add@amps 1017 \def\add@amp#1{\if m#1&\exp\add@amp\fi}
1018 \def\add@amps#1{%
1019   \begingroup
1020   \count@#1\advance\count@-\column@
1021   \edef\@tempa{\endgroup
1022     \exp\add@amp\romannumeral\number\count@ 000q}%
1023   \@tempa
1024 }
```

`\andhelp@` The help text stored in `\andhelp@` is used for errors generated by too many & characters in a row.

```

1025 \newhelp\andhelp@
1026 {An extra & here is so disastrous that you should probably exit^^J
1027 and fix things up.}

```

`\eqnshift@` `\eqnshift@` is used by `align` and `gather` as the indentation of the lines of the environment from the left margin.

```

1028 \newdimen\eqnshift@

```

`\alignsep@`

```

1029 \newdimen\alignsep@

```

`\tagshift@`

```

1030 \newdimen\tagshift@

```

`\mintagsep` `\mintagsep` is the minimum allowable separation between an equation and its tag. We set it to half a quad in `\textfont2`, which is $\text{T}_{\text{E}}\text{X}$'s built-in value.

```

1031 \newcommand{\mintagsep}{.5\fontdimen6\textfont\tw@}

```

`\minalignsep` This should probably be a skip register [mjd,1999/06/18]

```

1032 \newcommand{\minalignsep}{10pt}

```

`\tagwidth@`

```

1033 \newdimen\tagwidth@

```

`\totwidth@`

```

1034 \newdimen\totwidth@

```

`\lineht@` The dimen register `\lineht@` is used to keep track of the height (or depth, if tags are on the right) of a row in an alignment.

```

1035 \newdimen\lineht@

```

`\tag@width`

```

1036 \def\tag@width#1{%
1037     \ifcase\@xp#1\tag@lengths\fi
1038 }
1039
1040 \def\savetaglength@{%
1041     \begingroup
1042         \let\or\relax
1043         \xdef\tag@lengths{\tag@lengths\or \the\wdz@}%
1044     \endgroup
1045 }
1046
1047 \def\shift@tag#1{%
1048     \ifcase\@xp#1\tag@shifts\fi\relax
1049 }
1050
1051 \let\tag@shifts\@empty

```

`\saveshift@`

```

1052 \def\saveshift@#1{%
1053     \begingroup
1054         \let\or\relax
1055         \xdef\tag@shifts{\or#1\tag@shifts}%
1056     \endgroup
1057 }

```

`\spread@equation` This does the line-spacing adjustment that is normally wanted for displayed equations. We also call `\reset@strutbox@` here because otherwise a preceding font size change might leave `\strutbox@` with wrong contents. This is a less-than-ideal solution but probably good enough for now, until the situation can be overhauled.

```
1058 \def\spread@equation{\reset@strutbox@
1059   \openup\jot \let\spread@equation\@empty}
1060 \let\@spread@equation\spread@equation
```

`\display@` `\display@` is from `plain.tex`, with `\interdisplaylinepenalty` changed to `\eqpen`. Also we transplanted most of its internal organs to `\@display@init` to support `\display@` and other possibilities. Don't try to make sense of these naming conventions! They are a narrowly calculated mishmash of Knuth/Spi-vak/Lamport/Mittelbach precedents. The reason for not cleaning them up and forcing all names to a consistent scheme is that then in principle we'd have to do it everywhere else too. And we programmers are paranoid about the side effects of name changes.

```
1061 \def\display@{\@display@init{}}
1062 \def\@display@init#1{%
1063   \global\dt@ptrue \spread@equation
1064   \everycr{%
1065     \noalign{%
1066       #1%
1067       \ifdt@p
1068         \global\dt@pfalse
1069         \vskip-\lineskiplimit
1070         \vskip\normallineskiplimit
1071       \else
1072         \penalty\eqpen \global\dspbrk@lvl@m@ne
1073       \fi
1074     }%
1075   }%
1076 }
```

`\display@` is nearly the same; it additionally sets the `tag@` switch and the `\column@` and `\dspbrk@lvl` counters to their default values. The argument is normally a bit of code to empty out `\raise@tag`, but in `multline` we don't want that to happen in `\everycr`.

```
1077 \def\display@{\@display@init{%
1078   \global\column@\z@ \global\dspbrk@lvl@m@ne
1079   \global\tag@false \global\let\raise@tag\@empty
1080 }}
```

`\black@` This macro is made to produce an overflow box message and possibly (depending on the value of `\overfullrule`) a rule in the margin if the total width of an alignment is larger than the value of `\displaywidth`.

```
1081 \def\black@#1{%
1082   \noalign{%
1083     \ifdim#1>\displaywidth
1084       \dimen@\prevdepth
1085       \nointerlineskip
1086       \vskip-\ht\strutbox@
1087       \vskip-\dp\strutbox@
1088       \vbox{\noindent\hbox to#1{\strut@\hfill}}%
1089       \prevdepth\dimen@
1090     \fi
1091   }%
1092 }
```

`\savecounters@` These are used during the measuring phase of the various display math environments to save and restore the values of all L^AT_EX counters. We make these local to a group, so nested environments works.

Changed `\stepcounter` to `\csname c@... \endcsname` to avoid overhead of ifundefined test [mjd, 1995/01/20].

```

1093 \def\savecounters@{%
1094     \begingroup
1095     \def\@elt##1{%
1096         \global\csname c@##1\endcsname\the\csname c@##1\endcsname}%
1097     \xdef\@gtempa{%
1098         \cl@ckpt
1099         \let\@nx\restorecounters@\@nx\@empty
1100     }%
1101     \endgroup
1102     \let\restorecounters@\@gtempa
1103 }
1104 %
1105 \let\restorecounters@\@empty

```

`\savealignstate@` These are used to save the values of various parameters that are shared by `align` and `gather` when the former is used inside the latter.

```

1106 \def\savealignstate@{%
1107     \begingroup
1108     \let\or\relax
1109     \xdef\@gtempa{%
1110         \global\totwidth@\the\totwidth@
1111         \global\row@\the\row@
1112         \gdef\@nx\tag@lengths{\tag@lengths}%
1113         \let\@nx\restorealignstate@\@nx\@empty
1114     }%
1115     \endgroup
1116     \let\restorealignstate@\@gtempa
1117 }
1118
1119 \let\restorealignstate@\@empty

```

```

\savecolumn@
\restorecolumn@
1120 \def\savecolumn@{%
1121     \edef\restorecolumn@{%
1122         \global\column@\number\column@
1123         \let\@nx\restorecolumn@\@nx\@empty
1124     }%
1125 }
1126 \let\restorecolumn@\@empty

```

16.3 Scanning the environment's body

Several of the math alignment macros must scan their body twice: once to determine how wide the columns are and then to actually typeset them. This means that we must collect all text in this body before calling the environment macros.

`\@envbody` We start by defining a token register to contain the body.

```
1127 \newtoks\@envbody
```

`\addto@envbody` Then we define a macro to add something (i.e. its argument) to the token register `\@envbody`.

```
1128 \def\addto@envbody#1{\global\@envbody\exp{\the\@envbody#1}}
```

`\collect@body` The macro `\collect@body` starts the scan for the `\end{...}` command of the current environment. It takes a macro name as argument. This macro is supposed to take the whole body of the environment as its argument. For example, `\begin{align}` would call `\collect@body\@align` if `\@align#1{...}` is the macro that sets the alignment with body #1.

```
1129 \def\collect@body#1{%
1130   \@envbody{\@xp#1\@xp{\the\@envbody}}%
1131   \edef\process@envbody{\the\@envbody\@nx\end{\@currenvir}}%
1132   \@envbody\@emptytoks \def\begin@stack{b}%
```

If we simply called `\collect@@body` directly, the error message for a `\par` token (usually from a blank line) would be

```
! Paragraph ended before \collect@@body was complete.
```

But we use a little finesse to get a more intelligible error message:

```
! Paragraph ended before \multiline* was complete.
```

In order to avoid using up csnames unnecessarily we use the actual environment name as the name of the temporary function that is `\let` to `\collect@@body`; but then in order to preserve the theoretical possibility of nesting for environments that use `\collect@body` (not currently required by any `amsmath` environment [mjd,1999/06/23]), we do the `\let` inside a group.

```
1133 \begingroup
1134 \@xp\let\curname\@currenvir\endcurname\collect@@body
```

This small twist eliminates the need for `\expandafter's` in `\collect@@body`.

```
1135 \edef\process@envbody{\@xp\@nx\curname\@currenvir\endcurname}%
1136 \process@envbody
1137 }
```

`\push@begins` When adding a piece of the current environment's contents to `\@envbody`, we scan it to check for additional `\begin` tokens, and add a 'b' to the stack for any that we find.

```
1138 \def\push@begins#1\begin#2{%
1139   \ifx\end#2\else b\@xp\push@begins\fi
1140 }
```

`\collect@@body` `\collect@@body` takes two arguments: the first will consist of all text up to the next `\end` command, the second will be the `\end` command's argument. If there are any extra `\begin` commands in the body text, a marker is pushed onto a stack by the `\push@begins` function. Empty state for this stack means that we have reached the `\end` that matches our original `\begin`. Otherwise we need to include the `\end` and its argument in the material that we are adding to our environment body accumulator.

Historical Note: In a former implementation, the error messages resulting from a typo in the environment name were unsatisfactory, because it was matching of the environment name that was used to determine the end of our environment body, instead of counting begin-end pairs. Thanks to Lars Hellström for a suggestion that led to this improvement. [mjd,1999/06/23]

```
1141 \def\collect@@body#1\end#2{%
1142   \edef\begin@stack{\push@begins#1\begin\end \@xp@gobble\begin@stack}%
1143   \ifx\@empty\begin@stack
1144     \endgroup
```

```

1145   \@checkend{#2}%
1146   \addto@envbody{#1}%
1147   \else
1148     \addto@envbody{#1\end{#2}}%
1149   \fi
1150   \process@envbody % A little tricky! Note the grouping
1151 }

```

16.4 Simple aligning environments

`\math@cr@@@aligned` From `tabskip` we get an extra space of `minalignsep` after every second column; but when this falls at the right edge of the whole aligned structure, we need to cancel that space.

```

1152 \def\math@cr@@@aligned{%
1153   \ifodd\column@ \let\next@\empty
1154   \else \def\next@{&\kern-\alignsep@}%
1155   \fi
1156   \next@ \cr
1157 }

```

`\start@aligned` The `aligned` and `alignedat` environments are identical except that the latter takes a mandatory argument to specify the number of align structures, while the former allows any number of align structures automatically (the use of `alignedat` is deprecated). So, they will be defined in terms of `\start@aligned`, which will take two arguments. The first argument specifies the placement of the environments; it is either `c`, `t`, or `b`. The second is the number of align structures; a value of `-1` means that an arbitrary number are allowed.

```

1158 \newcommand{\start@aligned}[2]{%
1159   \RIfM@else
1160     \nonmatherr@{\begin{\@currenvir}}%
1161   \fi
1162   \savecolumn@ % Assumption: called inside a group

```

The `\null\` here is to keep the `\`, glue from causing the invocation of the clause in `TEX`'s built-in tag placement algorithm that can cause an equation to be shifted all the way over to the margin.

```

1163   \null\,%
1164   \if #1t\top \else \if#1b \vbox \else \vcenter \fi \fi \bgroup
1165     \maxfields@#2\relax
1166     \ifnum\maxfields@>\m@ne
1167       \multiply\maxfields@\tw@

```

Introduced new `\math@cr@@@` so we can provide standard error message for too many `&`'s in `alignedat`.

```

1168       \let\math@cr@@@\math@cr@@@alignedat
1169       \alignsep@\z@skip
1170     \else
1171       \let\math@cr@@@\math@cr@@@aligned
1172       \alignsep@\minalignsep
1173     \fi

```

Reset the meaning of `\`.

```

1174     \Let@ \chardef\dspbrk@context\@ne

```

Restore the default definition of `\tag` (error message), in case `aligned` is used inside, e.g., a `gather` environment that accepts `\tag`.

```

1175     \default@tag
1176     \spread@equation % no-op if already called

```

Finally we start the alignment itself. For `aligned` we add `\minalignsep` after every second column to mimic the behavior of `align`. For `alignedat` the user has to specify interalign space explicitly.

```

1177     \global\column@ \z@
1178     \ialign\bgroup
1179         &\column@plus
1180         \hfil
1181         \strut@
1182         $\m@th\displaystyle{##}$%
1183         \tabskip\z@skip
1184         &\column@plus
1185         $\m@th\displaystyle{{}##}$%
1186         \hfil
1187         \tabskip\alignsep@
1188         \crcr
1189 }
```

`\math@cr@@@aligned` `\math@cr@@@aligned` checks to make sure the user hasn't put in too many `&`s in `alignedat`. Since `alignedat` doesn't use `\displ@y@`, we also reset `\column@` here. Note that in `aligned`, `\column@` will increase without bound, since it never gets reset, but this is harmless.

```

1190 \def\math@cr@@@alignedat{%
1191     \ifnum\column@>\maxfields@
1192         \begingroup
1193             \measuring@false
1194             \@amsmath@err{Extra & on this line}%
1195             {\the\andhelp@}% "An extra & here is disastrous"
1196         \endgroup
1197     \fi
1198     \global\column@ \z@
1199     \cr
1200 }
```

`\alignsafe@testopt` Testing for an optional argument can be really, really tricky in certain complicated contexts. This we discovered by getting some bug reports for uses of `aligned`. So here is a safer form of L^AT_EX's `\@testopt` function.

```

1201 \def\alignsafe@testopt#1#2{%
1202     \relax\iffalse{\fi\ifnum`}=0\fi
1203     \@ifnextchar[%
1204         {\let\@let@token\relax \ifnum`{=\z@\fi\iffalse}\fi#1}%
1205         {\let\@let@token\relax \ifnum`{=\z@\fi\iffalse}\fi#1[#2]}%
1206 }
```

`aligned` The `aligned` environment takes an optional argument that indicates its vertical position in relation to surrounding material: `t`, `c`, or `b` for top, center, or bottom.

```

1207 \newenvironment{aligned}{%
1208     \let\@testopt\alignsafe@testopt
1209     \aligned@a
1210 }{%
1211     \crcr\egroup
1212     \restorecolumn@
1213     \egroup
1214 }
1215 \newcommand{\aligned@a}[1][c]{\start@aligned{#1}\m@ne}
```

`alignedat` To get a top or bottom positioned `alignedat` structure, you would write something like

```
\begin{alignedat}[t]{3}
```

```
1216 \newenvironment{alignedat}{%
1217   \let\@testopt\alignsafe@testopt
1218   \alignedat@a
1219 }{%
1220   \endaligned
1221 }
1222 \newcommand{\alignedat@a}[1][c]{\start@aligned{#1}}
```

gathered The `gathered` environment is for several lines that are centered independently.

```
1223 \newenvironment{gathered}[1][c]{%
1224   \RIfM@else
1225   \nonmatherr@{\begin{gathered}}%
1226   \fi
1227   \null\,%
1228   \if #1t\top \else \if #1b\bottom \else \vcenter \fi \fi \bgroup
1229   \Let@ \chardef\dspbrk@context\@ne \restore@math@cr
1230   \spread@equation
1231   \ialign\bgroup
1232     \hfil\strut@$\m@th\displaystyle##$\hfil
1233     \crcr
1234 }{%
1235   \endaligned
1236 }
```

16.5 The gather environment

`\start@gather`

```
1237 \def\start@gather#1{%
1238   \RIfM@
1239   \nomath@env
1240   \DN@{\@namedef{end\@currenvir}{}\@gobble}%
1241   \else
1242     $$%
1243     #1%
1244     \ifst@rred \else \global\@eqnswtrue \fi
1245     \let\next@\gather@
1246   \fi
1247   \collect@body\next@
1248 }
```

`gather`

```
gather* 1249 \newenvironment{gather}{%
1250   \start@gather\st@rredfalse
1251 }{%
1252   \math@cr \black@totwidth@ \egroup
1253   $$\ignorespacesafterend
1254 }
1255
1256 \newenvironment{gather*}{%
1257   \start@gather\st@rredtrue
1258 }{%
1259   \endgather
1260 }
```

`\gather@`

```
1261 \def\gather@#1{%
1262   \ingather@true \let\split\insplit@
```



```

1263 \let\tag\tag@in@align \let\label\label@in@display
1264 \chardef\dspbrk@context\z@
1265 \intertext@ \displ@y@ \Let@
1266 \let\math@cr@@@math@cr@@@gather
1267 \gmeasure@{#1}%
1268 \global\shifttag@false
1269 \tabskip\z@skip
1270 \global\row@ \@ne
1271 \halign to\displaywidth\bgroup
1272 \strut@
1273 \setboxz@h{\$ \m@th\displaystyle{##}$}%
1274 \calc@shift@gather
1275 \set@gather@field
1276 \tabskip \@centering
1277 &\setboxz@h{\strut@{##}}%
1278 \place@tag@gather
1279 \tabskip \iftagsleft@ \gdisplaywidth@ \else \z@skip \span\fi
1280 \crcr
1281 #1%
1282 }

```

\gmeasure@

```

1283 \def\gmeasure@#1{%
1284 \begin@group
1285 \measuring@true
1286 \totwidth@\z@
1287 \global\let\tag@lengths\@empty
1288 \savecounters@
1289 \setbox\@ne\vbox{%
1290 \everycr{\noalign{\global\tag@false
1291 \global\let\raise@tag\@empty \global\column@\z@}}%
1292 \let\label\@gobble
1293 \halign{%
1294 \setboxz@h{\$ \m@th\displaystyle{##}$}%
1295 \ifdim\wdz@>\totwidth@
1296 \global\totwidth@\wdz@
1297 \fi
1298 &\setboxz@h{\strut@{##}}%
1299 \savetaglength@
1300 \crcr
1301 #1%
1302 \math@cr@@@
1303 }%
1304 }%
1305 \restorecounters@
1306 \if@fleqn
1307 \global\advance\totwidth@\@mathmargin
1308 \fi
1309 \iftagsleft@
1310 \ifdim\totwidth@>\displaywidth
1311 \global\let\gdisplaywidth@\totwidth@
1312 \else
1313 \global\let\gdisplaywidth@\displaywidth
1314 \fi
1315 \fi
1316 \endgroup
1317 }

```

\math@cr@@@gather Modified \math@cr@@@gather so that it always puts in the final field, which

needs to be done under the new method for determining tag placement. This is probably more efficient anyway.

```

1318 \def\math@cr@@@gather{%
1319     \ifst@rred\nonumber\fi
1320     &\relax
1321     \make@display@tag
1322     \ifst@rred\else\global\@eqnswtrue\fi

```

We advance `\row@` here, rather than at the beginning of the preamble, because otherwise the `split` environment will cause `\row@` to be advanced twice instead of once.

```

1323     \global\advance\row@\@ne
1324     \cr
1325 }

```

`\calc@shift@gather` `\calc@shift@gather` has must make two decisions: (1) whether the equation tag for the current line should be put on a separate line and (2) what the distance between the equation and the equation tag should be. We implement \TeX 's built-in tag-placement as well as possible, with one improvement: the minimum separation between tag and equation is now a user-settable parameter.

[1995/01/17] Added a check to make sure that the width of the tag on the current line is > 0 before testing to see if $\text{tagwidth} + \text{linewidth} + \text{mintagsep} > \text{displaywidth}$. Since an imbedded align shows up as line with width `\displaywidth`, and even lines without a tag get processed as if an empty tag were present, the result was that the empty tag assigned to the line containing the align was being shifted downwards, creating extra space after the align.

```

1326 \def\calc@shift@gather{%
1327     \dimen@\mintagsep\relax
1328     \tagwidth@\tag@width\row@\relax

```

If we're in `fleqn` mode, there is no flexibility about placement of the equation, so all we can do is see if there's room for the tag in the given margin.

```

1329     \if@fleqn
1330         \global\eqnshift@\@mathmargin
1331         \ifdim\tagwidth@>\z@
1332             \advance\dimen@\tagwidth@
1333             \iftagsleft@
1334                 \ifdim\dimen@>\@mathmargin
1335                     \global\shifttag@true
1336             \fi
1337         \else
1338             \advance\dimen@\@mathmargin
1339             \advance\dimen@\wdz@
1340             \ifdim\dimen@>\displaywidth
1341                 \global\shifttag@true
1342             \fi
1343         \fi
1344     \fi
1345 \else
1346     \global\eqnshift@\displaywidth
1347     \global\advance\eqnshift@-\wdz@
1348     \ifdim\tagwidth@>\z@
1349         \multiply\dimen@\tw@
1350         \advance\dimen@\wdz@
1351         \advance\dimen@\tagwidth@
1352         \ifdim\dimen@>\displaywidth
1353             \global\shifttag@true
1354         \else

```

```

1355             \ifdim\eqnshift@<4\tagwidth@
1356             \global\advance\eqnshift@-\tagwidth@
1357             \fi
1358         \fi
1359     \fi
1360     \global\divide\eqnshift@\tw@
1361     \iftagsleft@
1362         \global\eqnshift@-\eqnshift@
1363         \global\advance\eqnshift@\displaywidth
1364         \global\advance\eqnshift@-\wdz@
1365     \fi
1366     \ifdim\eqnshift@<\z@
1367         \global\eqnshift@\z@
1368     \fi
1369 \fi
1370 }

```

\place@tag@gather

\set@gather@field

```

1371 \def\place@tag@gather{%
1372     \iftagsleft@
1373         \kern-\gdisplaywidth@
1374         \ifshifftag@
1375             \rlap{\vbox{%
1376                 \normalbaselines
1377                 \boxz@
1378                 \vbox to\lineht@{}}%
1379             \raise@tag
1380         }}%
1381         \global\shifftag@false
1382     \else
1383         \rlap{\boxz@}%
1384     \fi
1385 \else
1386     \ifdim\totwidth@>\displaywidth
1387         \dimen@\totwidth@
1388         \advance\dimen@-\displaywidth
1389         \kern-\dimen@
1390     \fi
1391     \ifshifftag@
1392         \llap{\vtop{%
1393             \raise@tag
1394             \normalbaselines
1395             \setbox\@ne\null
1396             \dp\@ne\lineht@
1397             \box\@ne
1398             \boxz@
1399         }}%
1400         \global\shifftag@false
1401     \else
1402         \llap{\boxz@}%
1403     \fi
1404 \fi
1405 }
1406 %
1407 \def\set@gather@field{%
1408     \iftagsleft@
1409         \global\lineht@\ht\z@
1410     \else
1411         \global\lineht@\dp\z@

```

```

1412 \fi
1413 \kern\eqnshift@
1414 \boxz@
1415 \hfil
1416 }

```

16.6 The align family of environments

The `align`, `flalign`, `alignat`, `xalignat`, and `xxalignat` environments are virtually identical, and thus will share much code. We'll refer to the environments generically as “align” and will distinguish between them explicitly only when necessary.

```

\ifxxat@ The \xatlevel@ macro will be used, informally speaking, to distinguish between
\ifcheckat@ the alignat and xalignat, and xxalignat environments.
\xatlevel@
1417 \newif\ifxxat@
1418
1419 \newif\ifcheckat@
1420
1421 \let\xatlevel@\@empty

```

```

\start@align \start@align will be called by all of the align-like environments. The first
argument will be the \xatlevel@, i.e., 0, 1, or 2; the second argument will
be either \st@rredtrue or \st@rredfalse. The third argument will be the
number of aligned structures in the environment (either as supplied by the
user, or -1 to indicate that checking shouldn't be done). After performing the
appropriate error detection and initialization, \start@align calls \align@.

```

Note that the `\equation` counter is no longer stepped at the beginning of these environments.

TODO: Implement `\shoveleft` and `\shoveright` for align.

```

1422 \def\start@align#1#2#3{%
1423   \let\xatlevel@#1% always \z@, \@ne, or \tw@
1424   \maxfields@#3\relax
1425   \ifnum\maxfields@>\m@ne
1426     \checkat@true
1427     \ifnum\xatlevel@=\tw@
1428       \xxat@true
1429     \fi
1430     \multiply\maxfields@\tw@
1431   \else
1432     \checkat@false
1433   \fi
1434   \ifingather@
1435     \iffalse{\fi\ifnum0=}\fi
1436     \DN@{\vcenter\bgroup\savealignstate@\align@#2}%
1437   \else
1438     \ifmmode
1439       \if@display
1440         \DN@{\align@recover}%
1441       \else
1442         \nomath@env
1443         \DN@{\@namedef{end@\currenenv}{}}\@gobble}%
1444       \fi
1445     \else
1446       $$%
1447       \let\split\insplit@
1448       \DN@{\align@#2}%
1449     \fi

```

```

1450     \fi
1451     \collect@body\next@
1452 }

```

With version 1.2 of `amsmath`, it was possible to use `align*` and relatives in certain wrong contexts without getting an error, e.g.

```

\begin{equation*}
\begin{align*}
...
\end{align*}
\end{equation*}

```

For backward compatibility we therefore give only a warning for this condition instead of a full error, and try to recover using the `aligned` environment. The alignment of the material may be adversely affected but it will at least remain readable.

```

1453 \def\align@recover#1#2#3{%
1454   \endgroup
1455   \@amsmath@err{%
1456     Erroneous nesting of equation structures;\MessageBreak
1457     trying to recover with `aligned'%
1458   }\@ehc
1459   \begin{aligned}\relax#1\end{aligned}%
1460 }

```

`align` The definitions of the various `align` environments are quite straight-forward.

```

align* 1461 \newenvironment{alignat}{%
flalign 1462   \start@align\z@\st@rredfalse
flalign* 1463 }{%
alignat 1464   \endalign
alignat* 1465 }
xalignat 1466 \newenvironment{alignat*}{%
xalignat* 1467   \start@align\z@\st@rredtrue
xxalignat 1468 }{%
1469   \endalign
1470 }
1471 \newenvironment{xalignat}{%
1472   \start@align\@ne\st@rredfalse
1473 }{%
1474   \endalign
1475 }
1476 \newenvironment{xalignat*}{%
1477   \start@align\@ne\st@rredtrue
1478 }{%
1479   \endalign
1480 }
1481 \newenvironment{xxalignat}{%
1482   \start@align\tw@\st@rredtrue
1483 }{%
1484   \endalign
1485 }
1486 \newenvironment{align}{%
1487   \start@align\@ne\st@rredfalse\m@ne
1488 }{%
1489   \math@cr \black@\totwidth@
1490   \egroup
1491   \ifingather@
1492     \restorealignstate@

```

```

1493 \egroup
1494 \nonumber
1495 \ifnum0=`{\fi\iffalse}\fi
1496 \else
1497   $$%
1498 \fi
1499 \ignorespacesafterend
1500 }
1501 \newenvironment{align*}{%
1502   \start@align\@ne\st@rredtrue\m@ne
1503 }{%
1504   \endalign
1505 }
1506 \newenvironment{flalign}{%
1507   \start@align\tw@\st@rredfalse\m@ne
1508 }{%
1509   \endalign
1510 }
1511 \newenvironment{flalign*}{%
1512   \start@align\tw@\st@rredtrue\m@ne
1513 }{%
1514   \endalign
1515 }

```

`\align@` TODO: Some of these sets of initializations show up in multiple places. It might be worth making an abbreviation for them.

```

1516 \def\align@#1#2{%
1517   \inalign@true \intertext@ \Let@ \chardef\dspbrk@context\z@
1518   \ifingather@ \else \disply@\fi
1519   \let\math@cr@@@ \math@cr@@@align
1520   \ifxxat@ \else \let\tag\tag@in@align \fi
1521   \let\label\label@in@display
1522   #1% set st@r
1523   \ifst@rred \else \global\@eqnswtrue \fi
1524   \measure@{#2}%
1525   \global\row@\z@
1526   \tabskip\eqnshift@
1527   \halign\bgroup
1528     \span\align@preamble\crcr
1529     #2%
1530 }

```

`\math@cr@@@align`

```

1531 \def\math@cr@@@align{%
1532   \ifst@rred \nonumber \fi
1533   \if@eqnsw \global\tag@true \fi
1534   \global\advance\row@\@ne
1535   \add@amps\maxfields@
1536   \omit
1537   \kern-\alignsep@
1538   \iftag@
1539     \setboxz@h{\@lign\strut@{\make@display@tag}}%
1540     \place@tag
1541   \fi
1542   \ifst@rred \else \global\@eqnswtrue \fi
1543   \global\lineht@\z@
1544   \cr
1545 }

```

`@cr@@@align@measure`

```

1546 \def\math@cr@@@align@measure{%
1547   &\omit
1548   \global\advance\row@ \@ne
1549   \ifst@rred\nonumber\fi
1550   \if@eqnsw \global\tag@true \fi
1551   \ifnum\column@>\maxfields@
1552     \ifcheckat@
1553       \begingroup
1554         \measuring@false
1555         \@amsmath@err{Extra & on this line}%
1556         {\the\andhelp@}% "An extra & here is disastrous"
1557       \endgroup
1558     \else
1559       \global\maxfields@\column@
1560     \fi
1561   \fi
1562   \setboxz@h{\@lign\strut@{%
1563     \if@eqnsw
1564       \stepcounter{equation}%
1565       \tagform@\theequation
1566     \else
1567       \iftag@\df@tag\fi
1568     \fi
1569   }}%
1570   \savetaglength@
1571   \ifst@rred\else\global\@eqnswtrue\fi
1572   \cr
1573 }
```

`\field@lengths`

`\savefieldlength@`

`\fieldlengths@`

```

1574 \let\field@lengths\@empty
1575
1576 \def\savefieldlength@{%
1577   \begingroup
1578     \let\or\relax
1579     \xdef\field@lengths{%
1580       \field@lengths
1581       \ifnum\column@=0
1582         \or
1583       \else
1584         ,%
1585       \fi
1586       \the\wdz@
1587     }%
1588   \endgroup
1589 }
1590
1591 \def\fieldlengths@#1{%
1592   \ifcase\@xp#1\field@lengths\fi
1593 }
```

`\maxcolumn@widths`

`\maxcolumn@widths` will be used to hold the widths of the fields of the `alignat` environment. The widths will be separated by the token `\or`, making it easy to extract a given width using `\ifcase`.

```
1594 \let\maxcolumn@widths\@empty
```

`\maxcol@width`

`\maxcol@width` n = maximum width of n th column of the current `alignat` (i.e., the n th field of `\maxcolumn@widths`.) It expands to a $\langle dimen \rangle$, so it can be used

as the right-hand side of a *variable assignment* or *arithmetic* statement. It's argument can be any *number*, *integer variable* or macro that expands to one of these. [Check to make sure this is true.]

This is subtler than it looks.

```
1595 \def\maxcol@width#1{%
1596     \ifcase\@xp#1\maxcolumn@widths\fi\relax
1597 }
```

Now comes the real fun. A typical `align` environments looks something like this, where the vertical bars mark the edges of the fields of the underlying `\halign`:

1	2	3	4	5	6	
$V_i + q_i v_j$	$= v_i,$	X_i	$= x_i - q_i x_j,$	U_i	$= u_i,$	for $i \neq j;$
V_j	$= v_j,$	X_j	$= x_j,$	U_j	$= u_j + \sum_{i \neq j} q_i u_i.$	(3)
						(4)

Note that each align structure consists of two fields, with no space between them (a small space has been added here to highlight the boundaries). Furthermore, the text inside the odd-numbered fields is flushright, while the text inside the even-numbered fields is flushleft. The equation tags (shown on the right here) can be on either the right or the left. If there is not room (in a sense to be defined shortly) for the tag on the same line as the equation, the tag will be shifted to a separate line.

Each environment also has a certain number of “flexible spaces,” meaning spaces whose width we are allowed to adjust to take up the amount of “free space” in the line, meaning the space not taken up by the equation tag and the fields of the underlying `\halign`.

The flexible spaces come in two flavors: interalign spaces and margin spaces. If there are n align structures ($n = 3$ in the illustration above), there are $n - 1$ interalign spaces, unless we are in an `alignat` environment, in which case there are no flexible interalign spaces.

The number of margin spaces is a little more complicated: Normally, there are two, but if we're in `fleqn` mode, there is only one. Furthermore, if we're in an `xxalignat` or `flalign` environment (corresponding to `\xatlevel@ = 2`, then there are no flexible margin spaces.

Calculating the interalign and margin spaces is done in two stages.

First, the total amount of free space is divided uniformly among all the flexible spaces, without regard for the lengths of the tags on the various lines. For the non-`fleqn` case, this corresponds to centering the align structures between the margins. Note that in `fleqn` mode, the right margin is still allowed to be larger than `\mathmargin`. This introduces an element of asymmetry into the appearance of the environment, but it has the advantage of leaving more space for equation tags in the right margin. If the right margin were constrained to be equal to the left margin in this case, tags would need to be shifted to a separate line more often than would be desirable.

Ordinarily, all flexible spaces will be given the same width. However, this is not invariably true, since the interalign spaces are constrained to be at least `\minalignsep` wide, while—in the absence of equation tags, at least—the margin spaces are allowed to shrink to zero. As we shall see in a minute, if there are tags in the environment, then the margins are also bounded below by `\mintagsep`.

Next, we examine each line of the environment that has a tag to see if there is a gap of at least `\mintagsep` between the equation and its tag. If there isn't, we attempt to center the equation between the tag and the opposite margin,

leaving a gap of at least `\mintagsep` on either side, in order to preserve some symmetry, i.e., we want the equation to *look* like it's centered between the margin and the tag, so we don't want the margin space to be less than the gap between the tag and the equation. (Arguably, it would be better to allow the margin space to shrink to zero in this case in order to avoid shifting the tag to a separate line at any cost, but that would require all of our calculations to be a little more complicated and hence a little slower.) Finally, if no values of the interalign spaces and the margins (with the constraints outlined above) will produce an acceptable distance between the equation and its tag, then the tag will be shifted to a separate line.

`\measure@` `\measure@` collects the various bits of information that we'll need to perform the calculations outlined above, namely, the number of align structures in the environment, the natural lengths of the fields on each row, the maximum widths of each column, and the widths of the equation tags on each line. It also calculates the number of flexible interalign and margin spaces and computes the initial values of the parameters `\eqnshift@` and `\alignsep@`, which correspond to the widths of the margins and the interalign spaces, respectively.

```

1598 \def\measure@#1{%
1599     \begingroup
1600     \measuring@true
1601     \global\eqnshift@z@
1602     \global\alignsep@z@
1603     \global\let\tag@lengths\@empty
1604     \global\let\field@lengths\@empty
1605     \savecounters@
1606     \global\setbox0\vbox{%
1607         \let\math@cr@@@math@cr@@@align@measure
1608         \everycr{\noalign{\global\tag@false
1609             \global\let\raise@tag\@empty \global\column@z@}}%
1610         \let\label\gobble
1611         \global\row@z@
1612         \tabskipz@
1613         \halign{\span\align@preamble\crcr
1614             #1%
1615             \math@cr@@@
1616             \global\column@z@
1617             \add@amps\maxfields@\cr
1618         }%
1619     }%
1620     \restorecounters@

```

It's convenient to have `\maxfields@` rounded up to the nearest even number, so that `\maxfields@` is precisely twice the number of align structures.

```

1621     \ifodd\maxfields@
1622         \global\advance\maxfields@\@ne
1623     \fi

```

It doesn't make sense to have a single align structure in either `flalign` or `xxalignat`. So, we check for that case now and, if necessary, switch to an `align` or `alignat`. Arguably, we should issue a warning message, but why bother?

```

1624     \ifnum\xatlevel@=\tw@
1625         \ifnum\maxfields@<\thr@@
1626             \let\xatlevel@\z@
1627         \fi
1628     \fi

```

`\box0` now contains the lines of the `\halign`. After the following maneuver, `\box1` will contain the last line of the `\halign`, which is what we're interested in. (Incidentally, the penalty we're removing is the `\@eqpen` inserted by `\math@cr`. Normally, this is `\interdisplaylinepenalty`, unless the user has overridden that with a `\displaybreak` command.)

```
1629      \setbox\z@\vbox{%
1630      \unvbox\z@ \unpenalty \global\setbox\@ne\lastbox
1631      }%
```

`\box1` begins with `\tabskip` glue and contains alternating `\hboxes` (the fields whose widths we're trying to get) and `\tabskip` glue [need better diagram]:

```
\hbox{\tabskip\hbox\tabskip...\hbox\tabskip}
```

In fact, all the `\tabskip` glue will be 0pt, because all the `\tabskip`s in an `alignat` environment have a natural width of 0pt, and the `\halign` has been set in its natural width.

One nice result of this is that we can read `\totwidth@` off immediately, since it is just the width of `\box1`, plus `\@mathmargin` if we're in `fleqn` mode. (Actually, we also have to take `\minalignsep` into account, but we'll do that later):

```
1632      \global\totwidth@\wd\@ne
1633      \if@fleqn \global\advance\totwidth@\@mathmargin \fi
```

Now we initialize `\align@lengths` and start peeling the boxes off, one by one, and adding their widths to `\align@lengths`. We stop when we run out of boxes, i.e., when `\lastbox` returns a void box. We're going to build a list using `\or` as a delimiter, so we want to disable it temporarily.

```
1634      \global\let\maxcolumn@widths\@empty
1635      \begingroup
1636      \let\or\relax
1637      \loop
1638      \global\setbox\@ne\hbox{%
1639      \unhbox\@ne \unskip \global\setbox\thr@@\lastbox
1640      }%
1641      \ifhbox\thr@@
1642      \xdef\maxcolumn@widths{ \or \the\wd\thr@@ \maxcolumn@widths}%
1643      \repeat
1644      \endgroup
```

Now we calculate the number of flexible spaces and the initial values of `\eqnshift@` and `\alignsep@`. We start by calculating `\displaywidth - \totwidth@`, which gives us the total amount of "free space" in a row.

```
1645      \dimen@\displaywidth
1646      \advance\dimen@-\totwidth@
```

Next we calculate the number of columns of flexible spaces in the display, which depends on whether we're in `fleqn` mode and in which particular environment we are in.

We use `\@tempcnta` to store the total number of flexible spaces in the align and `\@tempcntb` for the number of interalign spaces.

```
1647      \ifcase\xatlevel@
```

In `alignat`, the interalign spaces are under user control, not ours. So, we set `\alignsep@` and `\minalignsep` both equal to 0pt. Later, when calculating a new value for `\alignsep@`, we will only save the new value if it is less than the current value of `\alignsep@` (i.e., `\alignsep@` will never increase). Since the values we calculate will never be negative, this will ensure that `\alignsep@` remains zero in `alignat`.

```

1648      \global\alignsep@\z@
1649      \let\minalignsep\z@
1650      \@tempcntb\z@

```

In `fleqn` mode, the left margin—and hence the right margin in this case—is fixed. Otherwise, we divide the free space equally between the two margins.

```

1651      \if@fleqn
1652          \@tempcnta\@ne
1653          \global\eqnshift@\@mathmargin
1654      \else
1655          \@tempcnta\tw@
1656          \global\eqnshift@\dimen@
1657          \global\divide\eqnshift@\@tempcnta
1658      \fi
1659      \or

```

In an `align` or `xalignat` environment with n aligned structures, there are $n - 1$ interalign spaces and either 1 or 2 flexible margins, depending on whether we're in `fleqn` mode or not.

```

1660      \@tempcntb\maxfields@
1661      \divide\@tempcntb\tw@
1662      \@tempcnta\@tempcntb
1663      \advance\@tempcntb\m@ne

```

If we are in `fleqn` mode, we fix the left margin and divide the free space equally among the interalign spaces and the right margin.

```

1664      \if@fleqn
1665          \global\eqnshift@\@mathmargin
1666          \global\alignsep@\dimen@
1667          \global\divide\alignsep@\@tempcnta
1668      \else

```

Otherwise, we divide the free space equally among the interalign spaces and both margins.

```

1669          \global\advance\@tempcnta\@ne
1670          \global\eqnshift@\dimen@
1671          \global\divide\eqnshift@\@tempcnta
1672          \global\alignsep@\eqnshift@
1673      \fi
1674      \or

```

Finally, if we're in an `flalign` or `xxalignat` environment, there are no flexible margins and $n - 1$ flexible interalign spaces.

```

1675      \@tempcntb\maxfields@
1676      \divide\@tempcntb\tw@
1677      \global\advance\@tempcntb\m@ne
1678      \global\@tempcnta\@tempcntb
1679      \global\eqnshift@\z@
1680      \global\alignsep@\dimen@

```

If we're in `fleqn` mode, we need to add back the `\@mathmargin` that was removed when `\dimen@` was originally calculated above.

```

1681      \if@fleqn
1682          \global\advance\alignsep@\@mathmargin\relax
1683      \fi
1684      \global\divide\alignsep@\@tempcntb
1685      \fi

```

Now we make sure `\alignsep@` isn't too small.

```

1686      \ifdim\alignsep@<\minalignsep\relax
1687          \global\alignsep@\minalignsep\relax

```

```

1688      \ifdim\eqnshift@>\z@
1689      \if@fleqn\else
1690      \global\eqnshift@ \displaywidth
1691      \global\advance\eqnshift@-\totwidth@
1692      \global\advance\eqnshift@-\@tempcntb\alignsep@
1693      \global\divide\eqnshift@\tw@
1694      \fi
1695      \fi
1696      \fi
1697      \ifdim\eqnshift@<\z@
1698      \global\eqnshift@\z@
1699      \fi
1700      \calc@shift@align

```

Next, we calculate the value of `\tagshift@`. This is the glue that will be inserted in front of the equation tag to make sure it lines up flush against the appropriate margin.

```

1701      \global\tagshift@\totwidth@
1702      \global\advance\tagshift@\@tempcntb\alignsep@
1703      \if@fleqn
1704      \ifnum\xatlevel@=\tw@
1705      \global\advance\tagshift@-\@mathmargin\relax
1706      \fi
1707      \else
1708      \global\advance\tagshift@\eqnshift@
1709      \fi
1710      \iftagsleft@ \else
1711      \global\advance\tagshift@-\displaywidth
1712      \fi

```

Finally, we increase `\totwidth@` by an appropriate multiple of `\minalignsep`. If the result is greater than `\displaywidth`, it means that at least one line in the `align` is overfull and we will issue an appropriate warning message (via `\bl@ck`) at the end of the environment.

```

1713      \dimen@\minalignsep\relax
1714      \global\advance\totwidth@\@tempcntb\dimen@
1715      \ifdim\totwidth@>\displaywidth
1716      \global\let\displaywidth@\totwidth@
1717      \else
1718      \global\let\displaywidth@\displaywidth
1719      \fi
1720      \endgroup
1721 }

```

The code for calculating the appropriate placement of equation tags in the `align` environments is quite complicated and varies wildly depending on the settings of the `tagsleft@` and `@fleqn` switches. To minimize memory and hash space usage, we only define the variant appropriate for the current setting of those switches.

It would be worthwhile to examine this code more closely someday and see if it could be optimized any.

Tag placement when `\tagsleft@true`, `\@fleqntrue`. We begin with the version of `\calc@shift@align` appropriate for flush-left equations with tags on the left.

`\calc@shift@align` This is the simplest case. Since the left margin is fixed, in general the only thing to do is check whether there is room for the tag in the left margin. The only exception is that if `\eqnshift@ = 0pt`—meaning that we’re in a `flalign`

environment and this is the first line with a tag that we've encountered—then we set `\eqnshift@ = \@mathmargin` and recalculate `\alignsep@`. This is done by `\x@calc@shift@lf`.

```

1722 \iftagsleft@\if@fleqn
1723     \def\calc@shift@alignf%
1724         \global\let\tag@shifts\@empty
1725     \begingroup

```

`\@tempdima` is initialized to `\@mathmargin - \mintagsep`, which yields the maximum size of a tag that will not be shifted to another line.

```

1726         \@tempdima\@mathmargin\relax
1727         \advance\@tempdima-\mintagsep\relax

```

Now we examine each row in turn. If the width of the tag on the line is non-positive—meaning either that there is no tag or else that the user has forced it to have zero width—we mark the tag to remain unshifted. Otherwise, we call `\x@calc@shift@lf` to determine whether any adjustments need to be made to `\eqnshift@` and `\alignsep@`. Note the difference in treatment of zero-width tags between this code and \TeX 's built-in algorithm: here, a width of zero prohibits the tag from being shifted, while in \TeX 's built-in algorithm, a width of zero forces the tag to be shifted.

```

1728         \loop
1729             \ifnum\row@>0
1730                 \ifdim\tag@width\row@>\z@
1731                     \x@calc@shift@lf
1732                 \else
1733                     \saveshift@0%
1734                 \fi
1735                 \advance\row@\m@ne
1736         \repeat
1737     \endgroup
1738 }

```

`\x@calc@shift@lf` As mentioned above, `\x@calc@shift@lf` first checks to see if the current left margin is set to 0 and, if so, resets it to `\@mathmargin` and recalculates `\alignsep@`. Next, it checks whether the length of the current tag exceeds the previously calculated limit and, if so, marks the tag to be shifted to a separate line.

```

1739     \def\x@calc@shift@lf{%
1740         \ifdim\eqnshift@=\z@
1741             \global\eqnshift@\@mathmargin\relax
1742             \alignsep@\displaywidth
1743             \advance\alignsep@-\totwidth@
1744             \global\divide\alignsep@\@tempcntb
1745             \ifdim\alignsep@<\minalignsep\relax
1746                 \global\alignsep@\minalignsep\relax
1747             \fi
1748         \fi
1749         \ifdim\tag@width\row@>\@tempdima
1750             \saveshift@1%
1751         \else
1752             \saveshift@0%
1753         \fi
1754     }
1755 \fi\fi

```

Tag placement when `\tagsleft@false`, `\@fleqntrue`. Next we consider the case when equations are flush-left, but tags are on the right. This case

is somewhat more complicated than the previous one, since we can adjust the right margin by varying the inter-align separation. Thus, when a tag is found to be too close to its equation, we first attempt to decrease `\alignsep@` enough to move the equation off to an acceptable distance. Only if that would require a value of `\alignsep@` less than `\minalignsep` do we move the tag to a separate line.

`\calc@shift@align` This version of `\calc@shift@align` differs from the previous version only in calling `\x@calc@shift@rf` rather than `\x@calc@shift@lf`.

```

1756 \iftagsleft@\else\if@fleqn
1757   \def\calc@shift@align{%
1758     \global\let\tag@shifts\@empty
1759     \begingroup
1760       \loop
1761         \ifnum\row@>0
1762           \ifdim\tag@width\row@>\z@
1763             \x@calc@shift@rf
1764           \else
1765             \saveshift@0%
1766           \fi
1767           \advance\row@\m@ne
1768       \repeat
1769     \endgroup
1770   }
```

`\x@calc@shift@rf` To start, we need to know two quantities: the number of align structures in the current row and the “effective length” of the row, defined as the distance from the left margin to the right edge of the text assuming that `\eqnshift@` and `\alignsep@` are both 0. To get the number of align structures, we first count the number of columns by counting the number of entries in the `\fieldlengths@` for the current row. The effective length is calculated by `\x@rcalc@width` and put in the temporary register `\@tempdimc`, using `\@tempdimb` as an auxiliary variable.

```

1771   \def\x@calc@shift@rf{%
1772     \column@\z@
1773     \@tempdimb\z@
1774     \@tempdimc\z@
1775     \edef\@tempb{\fieldlengths@\row@}%
1776     \for\@tempa:=\@tempb\do{%
1777       \advance\column@\@ne
1778       \x@rcalc@width
1779     }%
1780   \begingroup
```

If there are n columns in the current row, then there are $\lfloor (n+1)/2 \rfloor$ align structures and $\lfloor (n-1)/2 \rfloor$ interalign spaces.

```

1781     \advance\column@\m@ne
1782     \divide\column@\tw@
```

If this is smaller than the maximum number of interalign spaces in the environment, then we need to reduce `\@tempcnta` (the total number of flexible spaces in the current line) by `\@tempcntb` — `\column@` and reset `\@tempcntb` to `\column@`.

```

1783     \ifnum\@tempcntb>\column@
1784       \advance\@tempcnta-\@tempcntb
1785       \advance\@tempcnta\column@
1786       \@tempcntb\column@
1787     \fi
```

Next, we add the width of the tag and the (fixed) left margin to the effective length calculated above. This can be used to calculate how much “free space” there is in the current line and thus how much leeway we have to increase the amount of space between the tag and the equation.

```

1788      \tagwidth@\tag@width\row@\relax
1789      \@tempdima\eqnshift@
1790      \advance\@tempdima\@tempdimc\relax
1791      \advance\@tempdima\tagwidth@

```

The first thing to check is whether the tag should be shifted to a separate line. To do this, we add the minimum interalign separation and the `\mintagsep` to the value of `\@tempdima` just calculated. This yields the minimum acceptable length of the current line. If that is greater than `\displaywidth`, we mark the tag to be calculated. Otherwise, we mark the tag to be kept on the same line and then check to see if the `\alignsep@` needs to be reduced to make room for the tag.

```

1792      \dimen@\minalignsep\relax
1793      \multiply\dimen@\@tempcntb
1794      \advance\dimen@\mintagsep\relax
1795      \advance\dimen@\@tempdima
1796      \ifdim\dimen@>\displaywidth
1797          \saveshift@1%
1798      \else
1799          \saveshift@0%

```

Now we perform essentially the same calculation, but using the current value of `\alignsep@` instead of `\minalignsep`. This gives the current length of the line. If this is greater than `\displaywidth`, we recalculate `\alignsep@` to make room for the tag.

```

1800      \dimen@\alignsep@\relax
1801      \multiply\dimen@\@tempcntb
1802      \advance\dimen@\@tempdima
1803      \advance\dimen@\tagwidth@
1804      \ifdim\dimen@>\displaywidth
1805          \dimen@\displaywidth
1806          \advance\dimen@-\@tempdima
1807          \ifnum\xatlevel@=\tw@
1808              \advance\dimen@-\mintagsep\relax
1809          \fi
1810          \divide\dimen@\@tempcnta
1811          \ifdim\dimen@<\minalignsep\relax
1812              \global\alignsep@\minalignsep\relax
1813          \else
1814              \global\alignsep@\dimen@
1815          \fi
1816      \fi
1817      \fi
1818      \endgroup
1819  }
1820 \fi\fi

```

Tag placement when `\tagsleft@false`, `\@fleqnfalse`. This is similar to the previous case, except for the added complication that both `\alignsep@` and `\eqnshift@` can vary, which makes the computations correspondingly more complicated.

```
\calc@shift@align
```

```
1821 \iftagsleft@\else\if@fleqn\else
```

```

1822 \def\calc@shift@align{%
1823   \global\let\tag@shifts\@empty
1824   \begingroup
1825     \loop
1826       \ifnum\row@>0
1827         \ifdim\tag@width\row@>\z@
1828           \x@calc@shift@rc
1829         \else
1830           \saveshift@0%
1831         \fi
1832         \advance\row@\m@ne
1833     \repeat
1834   \endgroup
1835 }

\x@calc@shift@rc

1836 \def\x@calc@shift@rc{%
1837   \column@\z@
1838   \@tempdimb\z@
1839   \@tempdimc\z@
1840   \edef\@tempb{\fieldlengths@\row@}%
1841   \@for\@tempa:=\@tempb\do{%
1842     \advance\column@\@ne
1843     \x@rcalc@width
1844   }%
1845   \begingroup
1846     \advance\column@\m@ne
1847     \divide\column@\tw@
1848     \ifnum\@tempcntb>\column@
1849       \advance\@tempcnta-\@tempcntb
1850       \advance\@tempcnta\column@
1851       \@tempcntb\column@
1852     \fi
1853     \tagwidth@\tag@width\row@\relax
1854     \@tempdima\@tempdimc
1855     \advance\@tempdima\tagwidth@
1856     \dimen@\minalignsep\relax
1857     \multiply\dimen@\@tempcntb
1858     \advance\dimen@\mintagsep\relax
1859     \ifnum\xatlevel@=\tw@ \else
1860       \advance\dimen@\mintagsep\relax
1861     \fi
1862     \advance\dimen@\@tempdima
1863     \ifdim\dimen@>\displaywidth
1864       \saveshift@1%
1865     \else
1866       \saveshift@0%
1867       \dimen@\eqnshift@
1868       \advance\dimen@\@tempdima
1869       \advance\dimen@\@tempcntb\alignsep@
1870       \advance\dimen@\tagwidth@
1871       \ifdim\dimen@>\displaywidth
1872         \dimen@\displaywidth
1873         \advance\dimen@-\@tempdima
1874         \ifnum\xatlevel@=\tw@
1875           \advance\dimen@-\mintagsep\relax
1876         \fi
1877         \divide\dimen@\@tempcnta
1878         \ifdim\dimen@<\minalignsep\relax

```



```

1879             \global\alignsep@\minalignsep\relax
1880             \eqnshift@\displaywidth
1881             \advance\eqnshift@-\@tempdima
1882             \advance\eqnshift@-\@tempcntb\alignsep@
1883             \global\divide\eqnshift@\tw@
1884         \else
1885             \ifdim\dimen@<\eqnshift@
1886                 \ifdim\dimen@<\z@
1887                     \global\eqnshift@\z@
1888                 \else
1889                     \global\eqnshift@\dimen@
1890                 \fi
1891             \fi
1892             \ifdim\dimen@<\alignsep@
1893                 \global\alignsep@\dimen@
1894             \fi
1895         \fi
1896     \fi
1897 \fi
1898 \endgroup
1899 }
1900 \fi\fi

```

\x@rcalc@width

```

1901 \iftagsleft@\else
1902     \def\x@rcalc@width{%
1903         \ifdim\@tempa > \z@
1904             \advance\@tempdimc\@tempdimb
1905             \ifodd\column@
1906                 \advance\@tempdimc\maxcol@width\column@
1907                 \@tempdimb\z@
1908             \else
1909                 \advance\@tempdimc\@tempa\relax
1910                 \@tempdimb\maxcol@width\column@
1911                 \advance\@tempdimb-\@tempa\relax
1912             \fi
1913         \else
1914             \advance\@tempdimb\maxcol@width\column@\relax
1915         \fi
1916     }
1917 \fi

```

Tag placement when \tagsleft@true, \@fleqnfalse.

\calc@shift@align

```

1918 \iftagsleft@\if@fleqn\else
1919     \def\calc@shift@align{%
1920         \global\let\tag@shifts\@empty
1921         \begingroup
1922             \loop
1923                 \ifnum\row@>\z@
1924                     \ifdim\tag@width\row@>\z@
1925                         \x@calc@shift@lc
1926                     \else
1927                         \saveshift@0%
1928                     \fi
1929                     \advance\row@\m@ne
1930                 \repeat
1931         \endgroup

```

1932 }

\x@calc@shift@lc

1933 \def\x@calc@shift@lc{%
1934 \column@z@

\@tempdima will (eventually) be set to the effective width of the current row, defined as the distance from the leftmost point of the current line to the end of the last field of the \halign, ignoring any intervening \tabskips, plus the width of the current tag. That is, it will be the width of the first non-empty field plus the sum of the maximum widths of all following fields, plus the tag width.

\@tempdimb will be the “indentation” of leftmost end of text, ignoring the \tabskip glue, i.e., it will be the sum of the maximum widths of any fields to the left of the first non-empty field, plus whatever empty space there is at the beginning of the first non-empty field.

1935 \@tempdima\z@ % ``width of equation''
1936 \@tempdimb\z@ % ``indent of equation''
1937 \edef\@tempb{\fieldlengths@\row@}%
1938 \@for\@tempa:=\@tempb\do{%
1939 \advance\column@\@ne
1940 \x@lcalc@width
1941 }%
1942 \begingroup
1943 \tagwidth@\tag@width\row@\relax

\@tempdima is now easy to calculate, since it is just \totwidth@-\@tempdimb+\tagwidth@.

1944 \@tempdima\totwidth@
1945 \advance\@tempdima-\@tempdimb
1946 \advance\@tempdima\tagwidth@

Next, we check to see whether there is room for both the equation and the tag on the same line, by calculating the minimum acceptable length of the current row and comparing that to \displaywidth. Note that here we use \@tempcntb, i.e., the number of interalign spaces after the first non-empty align structure.

1947 \dimen@\minalignsep\relax
1948 \multiply\dimen@\@tempcntb
1949 \advance\dimen@\mintagsep\relax
1950 \ifnum\xatlevel@=\tw@ \else
1951 \advance\dimen@\mintagsep\relax
1952 \fi
1953 \advance\dimen@\@tempdima

If the minimum acceptable width of the current line is greater than \displaywidth, we mark the current tag to be shifted to a separate line.

1954 \ifdim\dimen@>\displaywidth
1955 \saveshift@1%
1956 \else

Otherwise, the tag can stay on the same line as the equation, but we need to check whether it is too close to the equation. So, we calculate the distance between the left margin and the left side of the equation, using the current values of \eqnshift@ and \alignsep@. Note that we use \count@ here, not \@tempcntb, as above.

1957 \saveshift@0%
1958 \dimen@\alignsep@
1959 \multiply\dimen@\count@
1960 \advance\dimen@\eqnshift@
1961 \advance\dimen@\@tempdimb

If the left margin is less than twice the tag width, we calculate new values of `\eqnshift@` and `\alignsep@` to move the equation further away from the tag. In particular, we center the current line between its tag and the right margin. Note that although we later will need to transform `\dimen@` into a value suitable for use as `\eqnshift@`, for the time being it is more useful to think of it as the space separating the tag from the equation.

```

1962          \ifdim\dimen@<2\tagwidth@
1963              \dimen@=\displaywidth
1964              \advance\dimen@-\@tempdima
1965              \ifnum\xatlevel@=\tw@
1966                  \advance\dimen@-\mintagsep\relax
1967              \fi

```

In certain circumstances we will get a divide-by-zero error here unless we guard against it. Use of `\@tempcnta` is complicated, sometimes it is assigned globally, sometimes locally. Need to sort it out one of these days [mjd,2000/06/02].

```

1968          \ifnum\@tempcnta>\z@
1969              \divide\dimen@\@tempcnta
1970          \else \dimen@\z@
1971          \fi

```

As usual, we check to make sure we don't set `\alignsep@` smaller than `\minalignsep` and, in any case, that we don't replace `\alignsep@` by a larger value.

```

1972          \ifdim\dimen@<\minalignsep\relax
1973              \global\alignsep@\minalignsep\relax
1974              \dimen@=\displaywidth
1975              \advance\dimen@-\@tempdima
1976              \advance\dimen@-\@tempcntb\alignsep@
1977              \global\divide\dimen@\tw@
1978          \else
1979              \ifdim\dimen@<\alignsep@
1980                  \global\alignsep@\dimen@
1981              \fi
1982          \fi

```

Next, we calculate an appropriate value of `\eqnshift@`, assuming that `\dimen@` is the desired separation between the tag and equation of the current line. This means that we first need to adjust `\dimen@` if we're in an `flalign` environment.

```

1983          \ifnum\xatlevel@=\tw@
1984              \dimen@\mintagsep\relax
1985          \fi

```

Now we calculate the value of `\eqnshift@` needed to produce a separation of `\dimen@` between the equation tag and the beginning of the equation. To do this, we need the following equation to hold:

$$\text{\eqnshift@} + n\text{\alignsep@} + \text{\@tempdimb} = \text{\tagwidth@} + \text{\dimen@}$$

where $n = \text{\count@}$ is the number of interalign spaces before the first non-empty field of the current line.

```

1986          \advance\dimen@\tagwidth@
1987          \advance\dimen@-\@tempdimb
1988          \advance\dimen@-\count@\alignsep@

```

The value of `\eqnshift@` just calculated is the minimum acceptable value; thus, we save it only if it is larger than the current value.

```

1989          \ifdim\dimen@>\eqnshift@
1990              \global\eqnshift@\dimen@
1991          \fi

```

```

1992          \fi
1993      \fi
1994      \endgroup
1995  }

```

`\x@lcalc@width` This macro calculates the “indentation” of the current row, as defined above under the description of `\x@calc@shift@lc`. This macro is called for each field of the current line, with `\@tempa` set to the width of the current field. Ideally, the loop enclosing `\x@lcalc@width` would terminate as soon as `\@tempa` is non-zero, but that would be a bit tricky to arrange. Instead, we use `\@tempdima` as a flag to signal when we’ve encountered the first non-empty field.

```

1996      \def\x@lcalc@width{%
1997          \ifdim\@tempdima = \z@

```

If the current field is empty (i.e., `\@tempa = 0 pt`, then we increment `\@tempdimb` by the width of the current field). Otherwise, we set `\@tempdima = 1 pt` as a signal value and increment `\@tempdimb` by the width of whatever empty space there might be at the left of the current field.

```

1998          \ifdim\@tempa > \z@
1999              \@tempdima\p@
2000              \ifodd\column@
2001                  \advance\@tempdimb \maxcol@width\column@
2002                  \advance\@tempdimb-\@tempa
2003          \fi

```

In addition, we need to adjust the values of `\@tempcnta` and `\@tempcntb` to account for any empty align structures that might occur at the beginning of the current line. More specifically, we first set `\count@` equal to the number of interalign spaces preceding the current field (namely, $\lfloor (\text{column@} - 1)/2 \rfloor$), and then subtract `\count@` from both `\@tempcnta` and `\@tempcntb`. The rationale is that for the purposes of adjusting the spacing between the tag and the equation, the only flexible interalign spaces are those after the first non-empty align structure, so we need to treat those different from the ones before the first non-empty align structure.

```

2004          \count@\column@
2005          \advance\count@\m@ne
2006          \divide\count@\tw@
2007          \advance\@tempcnta-\count@
2008          \advance\@tempcntb-\count@
2009      \else
2010          \advance\@tempdimb \maxcol@width\column@\relax
2011      \fi
2012  \fi
2013 }
2014 \fi\fi

```

`\place@tag` `\place@tag` takes care of the placement of tags in the `align` environments.

```

2015 \def\place@tag{%
2016     \iftagsleft@
2017         \kern-\tagshift@
2018         \if1\shift@tag\row@\relax
2019             \rlap{\vbox{%
2020                 \normalbaselines
2021                 \boxz@
2022                 \vbox to\lineht@{}}%
2023             \raise@tag
2024         }}%
2025     \else

```

```

2026      \rlap{\boxz@}%
2027      \fi
2028      \kern\displaywidth@
2029      \else
2030      \kern-\tagshift@
2031      \if1\shift@tag\row@\relax

```

Added depth to correct vertical spacing of shifted equation tags.—dmj, 1994/12/29

```

2032      \llap{\vtop{%
2033          \raise@tag
2034          \normalbaselines
2035          \setbox\@ne\null
2036          \dp\@ne\lineht@
2037          \box\@ne
2038          \boxz@
2039      }}%
2040      \else
2041      \llap{\boxz@}%
2042      \fi
2043      \fi
2044 }

```

`\align@preamble`

```

2045 \def\align@preamble{%
2046     &\hfil
2047     \strut@
2048     \setboxz@h{\@lign$\m@th\displaystyle{##}$}%
2049     \ifmeasuring@\savefieldlength@\fi
2050     \set@field
2051     \tabskip\z@skip
2052     &\setboxz@h{\@lign$\m@th\displaystyle{{}##}$}%
2053     \ifmeasuring@\savefieldlength@\fi
2054     \set@field
2055     \hfil
2056     \tabskip\alignsep@
2057 }

```

`\set@field` `\set@field` increments the column counter, tracks the value of `\lineht@` and finally inserts the box containing the contents of the current field.

```

2058 \def\set@field{%
2059     \column@plus
2060     \iftagsleft@
2061         \ifdim\ht\z@>\lineht@
2062             \global\lineht@\ht\z@
2063         \fi
2064     \else
2065         \ifdim\dp\z@>\lineht@
2066             \global\lineht@\dp\z@
2067         \fi
2068     \fi
2069     \boxz@
2070 }

```

16.7 The split environment

`\split@err` A special error function for `split` to conserve main mem (at a cost of string pool/hash size.

```

2071 \edef\split@err#1{%
2072     \@nx\@amsmath@err{%
2073         \string\begin{split} won't work here%

```

```

2074 }{%
2075   \exp\@nx\csname
2076 Did you forget a preceding \string\begin{equation}?^^J%
2077 If not, perhaps the 'aligned' environment is what
2078 you want.\endcsname}%
2079 }

```

split If the **split** environment occurs inside **align** or **gather**, it can make use of the enclosing **halign**; if it is called inside a simple equation, we add an implicit ‘gather’ container.

```

2080 \newenvironment{split}{%
2081   \if@display
2082     \ifinner
2083       \exp\@xp\@xp\split@aligned
2084     \else
2085       \ifst@rred \else \global\@eqnswtrue \fi
2086     \fi
2087   \else \let\endsplit\@empty \exp\collect@body\exp\split@err
2088   \fi
2089   \collect@body\gather@split
2090 }{%
2091   \crr
2092   \egroup
2093   \egroup
2094   \iftagsleft@ \exp\lendsplit@ \else \exp\rendsplit@ \fi
2095 }

2096 \let\split@tag\relax % init

2097 \def\gather@split#1#2#3{%
2098   \exp\endgroup \reset@equation % math@cr will handle equation numbering
2099   \iftag@
2100     \toks@{\exp{\df@tag}%
2101     \edef\split@tag{%
2102       \gdef\@nx\df@tag{\the\toks@}%
2103       \global\@nx\tag@true \@nx\nonumber
2104     }%
2105   \else \let\split@tag\@empty
2106   \fi
2107   \spread@equation

```

The extra **vcenter** wrapper here is not really a good thing but without it there are compatibility problems with old documents that throw in some extra material between **\begin{equation}** and **\begin{split}** (for example, **\hspace{-1pc}** or **\left\{**). [mjd,1999/09/20]

```

2108   \vcenter\bgroup
2109     \gather@\split@tag \begin{split}#1\end{split}}%
2110   \def\endmathdisplay@a{%
2111     \math@cr \black@ \totwidth@ \egroup
2112     \egroup
2113   }%
2114 }

```

\insplit@

```

2115 \def\insplit@{%
2116   \global\setbox\z@\vbox\bgroup
2117   \let@ \chardef\dspbrk@context\@ne \restore@math@cr
2118   \default@tag % disallow use of \tag here
2119   \ialign\bgroup
2120     \hfil

```

```

2121     \strut@
2122     $\m@th\displaystyle{##}$%
2123     &$\m@th\displaystyle{{}##}$%
2124     \hfill % Why not \hfil?---dmj, 1994/12/28
2125     \crrc
2126 }

```

`\rendsplit@` Moved the box maneuvers inside the `\ifinalign@`, since that is the only place they are needed.—dmj, 1994/12/28

TODO: Explore interaction of tag-placement algorithm with `split`. Is there any way for `split` to pass the relevant information out to the enclosing `gather` or `align`?

```

2127 \def\rendsplit@{%
2128     \ifinalign@

```

Changed `\box9` into a `\vtop` here for better spacing.

```

2129         \global\setbox9 \vtop{%
2130             \unvcopy\z@
2131             \global\setbox8 \lastbox
2132             \unskip
2133         }%
2134         \setbox\@ne\hbox{%
2135             \unhcopy8
2136             \unskip
2137             \global\setbox\tw@\lastbox
2138             \unskip
2139             \global\setbox\thr@@\lastbox
2140         }%
2141         \ifctagsplit@
2142             \gdef\split@{%
2143                 \hbox to\wd\thr@@{%
2144                     &\vcenter{\vbox{\moveleft\wd\thr@@\boxz}}}%
2145                 }%
2146             \else
2147                 \global\setbox7 \hbox{\unhbox\tw@\unskip}%

```

Added `\add@amps` to make sure we put the last line of the `split` into the proper column of an `align` environment with multiple align structures.—dmj, 1994/12/28

Special care has to be taken in this case because the `split` turns into two lines of the `align` instead of just one. So, we have to make sure that the first line produced by the `split` doesn't upset our bookkeeping, hence we call `\savetaglength@` to insert 0pt as the tag for this pseudo-line, and we advance the `\row@` counter and reset `\lineht@` afterwards. It would be nice if we could just replace the `\crrc` by `\math@cr@@@`, but that would cause problems with the tag processing.

```

2148         \gdef\split@{%
2149             \global\@tempcnta\column@
2150             &\setboxz@h{%
2151                 \savetaglength@
2152                 \global\advance\row@\@ne
2153                 \vbox{\moveleft\wd\thr@@\box9}%
2154                 \crrc
2155                 \noalign{\global\lineht@\z@}%
2156                 \add@amps\@tempcnta
2157                 \box\thr@@
2158             }%
2159         \fi
2160

```

```

2161 \else
2162   \ifctagsplit@
2163   \gdef\split@{\vcenter{\boxz@}}%
2164 \else

```

Changed to just `\boxz@`, otherwise last line gets centered rather than aligned properly with respect to the rest of the lines. But this means that we can't see inside of the last line to decide whether the tag needs to be moved. Will have to think about this.—dmj, 1994/12/28

```

2165   \gdef\split@{%
2166     \boxz@
2167 %     \box9
2168 %     \crrr
2169 %     \hbox{\box\thr@@\box7}}%
2170   }%
2171 \fi
2172 \fi
2173 \aftergroup\split@
2174 }

```

`\lendsplit@`

```

2175 \def\lendsplit@{%
2176   \global\setbox9\vtopy{\unvcopy\z@}%
2177   \iffinalign@

```

Moved following two boxes inside the `\iffinalign@`, since they are only used in that case. In fact, if we just kept track of the width of the first column, we could dispense with this entirely. Surely that would be more efficient than all these box copies.—dmj, 1994/12/28

```

2178   \setbox\@ne\vbox{%
2179     \unvcopy\z@
2180     \global\setbox8\lastbox
2181   }%
2182   \setbox\@ne\hbox{%
2183     \unhcopy8%
2184     \unskip
2185     \setbox\tw@\lastbox
2186     \unskip
2187     \global\setbox\thr@@\lastbox
2188   }%
2189   \ifctagsplit@
2190   \gdef\split@{%
2191     \hbox to\wd\thr@@{%
2192       &\vcenter{\vbox{\moveleft\wd\thr@@\box9}}}%
2193     }%
2194   \else
2195   \gdef\split@{%
2196     \hbox to\wd\thr@@{%
2197       &\vbox{\moveleft\wd\thr@@\box9}}%
2198     }%
2199   \fi
2200 \else
2201   \ifctagsplit@
2202   \gdef\split@{\vcenter{\box9}}%
2203   \else
2204   \gdef\split@{\box9}%
2205   \fi
2206 \fi
2207 \aftergroup\split@
2208 }

```


With `amsmath` 1.2 it was possible to put things like `\left\{` between `\begin{equation}` and `\begin{split}` without getting any error message. For backward compatibility we try to avoid a fatal error in this case and instead attempt recovery with `aligned`.

```
2209 \def\split@aligned#1#2{%
2210   \iffalse{\fi\ifnum0=}\fi
2211   \collect@body\split@al@a}

2212 \def\split@al@a#1#2#3{%
2213   \split@warning
2214   \endgroup
```

If the `fleqn` and `tbtags` options are both in effect then we will need to add an optional argument on the `aligned` environment.

```
2215   \toks@{\begin{aligned}}%
2216   \if@fleqn \split@al@tagcheck \fi
```

The `\relax` here is to prevent `\@let@token` from being left equal to an ampersand if that happens to be the first thing in the body.

```
2217   \the\toks@\relax#1\end{aligned}%
2218   \ifnum0= {\fi\iffalse}\fi
2219 }
```

```
2220 \def\split@al@tagcheck{%
2221   \ifctagsplit@
2222   \else
2223     \iftagsleft@ \toks@{\xp{\the\toks@ [t]}}%
2224     \else \toks@{\xp{\the\toks@ [b]}}%
2225     \fi
2226   \fi
2227 }

2228 \def\split@warning{%
2229   \PackageWarning{amsmath}{%
2230 Cannot use `split' here;\MessageBreak trying to recover with `aligned'}%
2231 }
```

16.8 The multiline environment

In the original $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$, `\multilinegap` is a macro with an argument that resets an internal dimension (one with an `@` character in its name). Here, to save control sequence names, we define `\multilinegap` to be the dimension itself and the documentation instructs users to use `\setlength` if they need to change it.

`\multilinegap` Changed `\multilinegap` and `\multlinetaggap` to skip registers. Also changed
`\multlinetaggap` name to `\multlinetaggap` from `\multlinetaggap@`.

```
2232 \newskip\multilinegap
2233 \multilinegap10pt
2234 \newskip\multlinetaggap
2235 \multlinetaggap10pt
```

`\start@multiline`

```
2236 \def\start@multiline#1{%
2237   \RIfM@
2238   \nomath@env
2239   \DN@{\@namedef{end\@currenvir}}{\@gobble}%
2240   \else
2241     $$%
2242     #1%
2243     \ifst@rred
```

```

2244         \nonumber
2245     \else
2246         \global\@eqnswtrue
2247     \fi
2248     \let\next@\multline@
2249 \fi
2250 \collect@body\next@
2251 }

\multline@
2252 \newenvironment{multline}{%
2253   \start@multline\st@rredfalse
2254 }{%
2255   \iftagsleft@ \exp\lendumultline@ \else \exp\rendmultline@ \fi
2256   \ignorespacesafterend
2257 }
2258 \newenvironment{multline*}{\start@multline\st@rredtrue}{\endumultline}

2259 \def\multline@#1{%
2260   \Let@
    For multiline neither \disply no \display@ is quite right; we want to
    advance the row number and (I suppose?) the display-pagebreak level, but we
    only want to do tag-related stuff once before the first line, not repeat it for every
    line. (Recall that the arg of \@display@init goes into \everycr.)
2261   \@display@init{\global\advance\row@\@one \global\dspbrk@lvl\m@ne}%
2262   \chardef\dspbrk@context\z@
2263   \restore@math@cr
    The multiline environment is somewhat unusual, in that \tag and \label are
    enabled only during the measuring phase and disabled during the production
    phase. Here we disable \tag and \label; \mmeasure@ will re-enable them
    temporarily.
2264   \let\tag\tag@in@align
2265   \global\tag@false \global\let\raise@tag\@empty
2266   \mmeasure@{#1}%
2267   \let\tag\gobble@tag \let\label\@gobble
2268   \tabskip \if@fleqn \@mathmargin \else \z@skip \fi
2269   \totwidth@\displaywidth
2270   \if@fleqn
2271     \advance\totwidth@-\@mathmargin
2272   \fi
2273   \halign\bgroup
2274     \hbox to\totwidth@{%
    In order to get the spacing of the last line right in fleqn mode, we need to
    play a little game here. Normally the stretchability of the \hskip here will be
    suppressed by the \hfil at the end of the template, except inside the last line,
    when that \hfil will be removed by the \hfilneg in \lendumultline@.
2275     \if@fleqn
2276       \hskip \@centering \relax
2277     \else
2278       \hfil
2279     \fi
2280     \strut@
2281     $\m@th\displaystyle{ }##\endumultline@math
2282     \hfil
2283   }%
2284   \crrc

```

In `fleqn` mode, it's the `\tabskip` of `\@mathmargin` that needs to be removed in the first line, not the `\hfil` at the beginning of the template.

```

2285     \if@fleqn
2286         \hskip-\@mathmargin
2287         \def\multline@indent{\hskip\@mathmargin}% put it back
2288     \else
2289         \hfilneg
2290         \def\multline@indent{\hskip\multlinegap}%
2291     \fi
2292     \iftagsleft@
2293         \iftag@
2294             \begingroup
2295                 \ifshifftag@
2296                     \rlap{\vbox{%
2297                         \normalbaselines
2298                         \hbox{%
2299                             \strut@
2300                             \make@display@tag
2301                         }%
2302                         \vbox to\lineht@{}%
2303                         \raise@tag
2304                     }}%

```

If the equation tag doesn't fit on the same line with the first line of the display, we'll indent the first line by `\multlinegap`. This is a change from `amstex`, where the first line would have been flush against the left margin in this case. A corresponding change will be made in `\rendmultline@`.

```

2305         \multline@indent
2306     \else
2307         \setbox\z@\hbox{\make@display@tag}%
2308         \dimen@\@mathmargin \advance\dimen@-\wd\z@
2309         \ifdim\dimen@<\multlinetaggap
2310             \dimen@\multlinetaggap
2311         \fi
2312         \box\z@ \hskip\dimen@\relax
2313     \fi
2314     \endgroup
2315 \else
2316     \multline@indent
2317 \fi
2318 \else
2319     \multline@indent
2320 \fi
2321 #1%
2322 }

```

An extra level of indirection for the closing `$` in `multline` allows us to avoid getting an extra `thinmuskip` from a final `mathpunct` in the equation contents, when equation numbers are on the right. If we did not use this workaround, the sequence of elements for a final comma would be, e.g.,

... ,<hskip><box containing equation number>

which is equivalent to a sequence `<mathpunct><mathord>` as far as the automatic math spacing is concerned.

```

2323 \def\endmultline@math{${}

```

`\lendmultline@` Bug fix: changed `\cr` to `\math@cr` so that `\@eqpen` gets reset properly if `\displaybreak` is used on the penultimate line of an `align`.

```

2324 \def\lendmultline{%
2325     \hfilneg
2326     \hskip\multlinegap
2327     \math@cr
2328     \egroup
2329     $$%
2330 }

```

\rendmultline@

```

2331 \def\rendmultline{%
2332     \iftag@
2333     $\let\endmultline\math\relax
2334     \ifshifftag@
2335     \hskip\multlinegap

```

Added depth to correct vertical spacing of shifted equation tags.—dmj, 1994/12/29

```

2336         \llap{\vtop{%
2337             \raise@tag
2338             \normalbaselines
2339             \setbox\@ne\mbox{
2340                 \dp\@ne\lineht@
2341                 \box\@ne
2342                 \hbox{\strut@make@display@tag}%
2343             }}%
2344         \else
2345             \hskip\multlinetaggap
2346             \make@display@tag
2347         \fi
2348     \else
2349         \hskip\multlinegap
2350     \fi
2351     \hfilneg

```

Use \math@cr rather than just \cr so that \eqpen gets reset properly if \displaybreak is used.

```

2352     \math@cr
2353     \egroup$$%
2354 }

```

\mmeasure@

```

2355 \def\mmeasure@#1{%
2356     \begingroup
2357     \measuring@true

```

We use \begin/endgroup rather than {} in this definition of \label because the latter would create an extra (wasteful of main mem) null box in the current math list. [mjd, 1995/01/17]

```

2358     \def\label##1{%
2359         \begingroup\measuring@false\label@in@display{##1}\endgroup}%
2360     \def\math@cr@@@{\cr}%
2361     \let\shoveleft\@iden \let\shoveright\@iden
2362     \savecounters@
2363     \global\row@ \z@
2364     \setbox\@ne\mbox{
2365         \global\let\df@tag\@empty
2366         \halign{
2367             \setboxz@h{\@align$\m@th\displaystyle{##1}$}%
2368             \iftagsleft@
2369                 \ifnum\row@=\@ne
2370                     \global\totwidth@\wdz@

```

```

2371          \global\lineht@ht\z@
2372          \fi
2373          \else
2374          \global\totwidth@wdz@
2375          \global\lineht@dp\z@
2376          \fi
2377          \crrc
2378          #1%
2379          \crrc
2380      }%
2381  }%
2382  \ifx\df@tag\@empty\else\global\tag@true\fi
2383  \if@eqnsw\global\tag@true\fi
2384  \iftag@
2385      \setboxz@h{%
2386          \if@eqnsw
2387              \stepcounter{equation}%
2388              \tagform@\theequation
2389          \else
2390              \df@tag
2391          \fi
2392      }%
2393      \global\tagwidth@wdz@
2394      \dimen@\totwidth@
2395      \advance\dimen@\tagwidth@
2396      \advance\dimen@\multlinetaggap
2397      \iftagsleft@\else
2398          \if@fleqn
2399              \advance\dimen@\@mathmargin
2400          \fi
2401          \fi
2402          \ifdim\dimen@>\displaywidth
2403              \global\shifftag@true
2404          \else
2405              \global\shifftag@false
2406          \fi
2407      \fi
2408      \restorecounters@
2409  \endgroup
2410 }

```

`\shoveleft` and `\shoveright` need to do slightly different things depending on whether tags are on the left or the right and whether we're in `fleqn` mode. For compactness of code, we make the appropriate decisions at “compile” time rather than at load time.

TODO: Investigate making `\shoveright` behave “properly”(?) if used on the first line of a `multline` and make `\shoveleft` behave properly if used on the last line of a `multline`. But in his `amstex.doc` Spivak indicates those commands should never be used on a first or last line. Perhaps better to leave the question open unless/until real-life examples turn up.

```

2411 \iftagsleft@
2412     \def\shoveright#1{%
2413         #1%
2414         \hfilneg
2415         \hskip\multlinegap
2416     }
2417 \else
2418     \def\shoveright#1{%
2419         #1%

```

```

2420     \hfilneg
2421     \iftag@
2422         \ifshiffttag@
2423             \hskip\multlinegap
2424         \else
2425             \hskip\tagwidth@
2426             \hskip\multlinetaggap
2427         \fi
2428     \else
2429         \hskip\multlinegap
2430     \fi
2431 }
2432 \fi
2433
2434 \if@fleqn
2435     \def\shoveleft#1{#1}%
2436 \else
2437     \iftagsleft@
2438         \def\shoveleft#1{%
2439             \setboxz@h{$\m@th\displaystyle{ }#1$}%
2440             \setbox\@ne\hbox{$\m@th\displaystyle#1$}%
2441             \hfilneg
2442             \iftag@
2443                 \ifshiffttag@
2444                     \hskip\multlinegap
2445                 \else
2446                     \hskip\tagwidth@
2447                     \hskip\multlinetaggap
2448                 \fi
2449             \else
2450                 \hskip\multlinegap
2451             \fi
2452             \hskip.5\wd\@ne
2453             \hskip-.5\wdz@
2454             #1%
2455         }
2456     \else
2457         \def\shoveleft#1{%
2458             \setboxz@h{$\m@th\displaystyle{ }#1$}%
2459             \setbox\@ne\hbox{$\m@th\displaystyle#1$}%
2460             \hfilneg
2461             \hskip\multlinegap
2462             \hskip.5\wd\@ne
2463             \hskip-.5\wdz@
2464             #1%
2465         }
2466     \fi
2467 \fi

```

16.9 The equation environment

Rewritten from the ground up for version 2.0 to fix no-shrink and no-shortskips bugs [mjd,2000/01/06].

Standard L^AT_EX provides three environments for one-line equations: `\[`, `\]`, `equation`, and `displaymath`. We add `equation*` as a synonym for `displaymath`.

```

2468 \@saveprimitive\leqno\@@leqno
2469 \@saveprimitive\eqno\@@eqno
2470 \def\eqno{\@@eqno\let\eqno\relax\let\leqno\relax}

```

```

2471 \def\leqno{\@@leqno\let\leqno\relax\let\eqno\relax}
2472 %
2473 \let\veqno=\@@eqno
2474 \iftagsleft@ \let\veqno=\@@leqno \fi

```

Support for the `showkeys` package: provide no-op definitions for a couple of SK functions, if they are not already defined. Then we can just call them directly in our code without any extra fuss. If the `showkeys` package is loaded later, our trivial definitions will get overridden and everything works fine.

```

2475 \ifundefined{SK@@label}{%
2476   \let\SK@@label\relax \let\SK@equationtrue\relax
2477 }{}

```

```
2478 \let\reset@equation\@empty
```

Cf `\tag@in@align`. This is a bit of a mess though. Could use some work. [mjd,1999/12/21]

```

2479 \let\alt@tag\@empty
2480 \def\tag@in@display#1#{\relax\tag@in@display@a{#1}}
2481 \def\tag@in@display@a#1#2{%
2482   \iftag@
2483     \invalid@tag{Multiple \string\tag}\relax
2484   \else
2485     \global\tag@true \nonumber \reset@equation \st@rredtrue
2486     \if * \string#1%
2487       \gdef\alt@tag{\def\SK@tagform@{#2\@gobble}}%
2488       \ifx\SK@@label\relax \let\tagform@\SK@tagform@ \fi
2489     }%
2490     \make@df@tag@{#2}%
2491   \else
2492     \make@df@tag@@{#2}%
2493   \fi
2494 \fi
2495 }

```

```
2496 \let\restore@hfuzz\@empty
```

```

2497 \def\mathdisplay#1{%
2498   \ifmmode \@badmath
2499   \else
2500     $$\def\@currenvir{#1}%

```

Allow use of `\displaybreak`.

```
2501   \let\dspbrk@context\z@
```

Although in some cases simpler label handling would seem to be sufficient, always using `\label@in@display` makes it easier to support the `showkeys` package.

```

2502   \let\tag\tag@in@display \let\label\label@in@display \SK@equationtrue
2503   \global\let\df@label\@empty \global\let\df@tag\@empty
2504   \global\tag@false
2505   \let\mathdisplay@push\mathdisplay@@push
2506   \let\mathdisplay@pop\mathdisplay@@pop
2507   \if@fleqn

```

Turn off overfull box messages temporarily—otherwise there would be unwanted extra ones emitted during our measuring operations.

```

2508     \edef\restore@hfuzz{\hfuzz\the\hfuzz\relax}%
2509     \hfuzz\maxdimen

```

Initially set the equation body in a box of displaywidth. Then if the box is not overfull, as we find by checking `\badness`, we have acquired useful information for the subsequent processing.

```

2510      \setbox\z@\hbox to\displaywidth\bgroup
2511      \let\split@warning\relax \restore@hfuzz
2512      \everymath\@emptytoks \m@th $\displaystyle
2513      \fi
2514      \fi
2515 }

```

Arg 1 is not currently used. I thought it might come in handy for error messages.

```

2516 \def\endmathdisplay#1{%
2517   \ifmmode \else \@badmath \fi
2518   \endmathdisplay@a
2519   $$$

```

I guess the following code means this structure is non-reentrant. But there is plenty of scope for tricky bugs here; suppressing them by brute force at least makes it possible to get things working correctly for normal use. [mjd,2000/01/06]

```

2520 \global\let\df@label\@empty \global\let\df@tag\@empty
2521 \global\tag@false \global\let\alt@tag\@empty
2522 \global\@eqnswfalse
2523 }

2524 \def\endmathdisplay@a{%
2525   \if@eqnsw \gdef\df@tag{\tagform@theequation}\fi
2526   \if@fleqn \exp\endmathdisplay@fleqn
2527   \else \ifx\df@tag\@empty \else \veqno \alt@tag \df@tag \fi
2528   \ifx\df@label\@empty \else \exp\ltx@label\exp{\df@label}\fi
2529   \fi
2530   \ifnum\dspbrk@lvl>\m@ne
2531     \postdisplaypenalty -\@getpen\dspbrk@lvl
2532     \global\dspbrk@lvl\m@ne
2533   \fi
2534 }

```

A boolean variable: Was that last box overfull or not? A value of 0 means yes, it was overfull.

```

2535 \let\too@wide\@ne

```

Special handling is needed for flush-left equations. We need to measure the equation body (found in box 0 after we close it with the `\egroup`). Then after a fairly normal test to see if it fits within the available space, we need to consider overlapping into the displayindent area if displayindent is nonzero (as in an indented list). If there is an equation number we may have to shift it by hand to a separate line when there is not enough room; we can no longer take advantage of the automatic shifting provided by the `\leqno`, `\eqno` primitives.

We initially add `\@mathmargin` glue at the end of box 0 to get an accurate overfull test. If `\@mathmargin` contains any shrink then we cannot reliably tell whether the box will be overfull or not simply by doing hand calculations from the actual width of the equation body. We have to actually set the box and find out what happens.

On the other hand if we put the `\@mathmargin` glue at the beginning of the box it's awkward to remove it afterwards. So we first put it in at the end and later we will move it to the beginning as needed.

```

2536 \def\endmathdisplay@fleqn{%
2537   $\hfil\hskip\@mathmargin\egroup

```

We need to save the information about whether box 0 was overfull in a variable, otherwise it will disappear in the next setbox operation. And we couldn't set the equation number box earlier than now, because the body of the equation

might have contained a `\tag` command (well, it could have been done, but this way we can reuse the tag-handling code from elsewhere).

```

2538 \ifnum\badness<\inf@bad \let\too@wide\@ne \else \let\too@wide\z@ \fi
2539 \ifx\@empty\df@tag
2540 \else
2541 \setbox4\hbox{\df@tag
2542 \ifx\df@label\@empty \else \@xp\ltx@label\@xp{\df@label}}\fi
2543 }%
2544 \fi
2545 \csname emdf@%
2546 \ifx\df@tag\@empty U\else \iftagsleft@ L\else R\fi\fi
2547 \endcsname
2548 }
```

For an unnumbered flush-left equation we hope first that the contents fit within `displaywidth`. If not we need to fall back on a more complicated reboxing operation.

```

2549 \def\emdf@U{%
2550 \restore@hfuzz
2551 \ifodd\too@wide % not too wide: just need to swap the glue around
2552 \hbox to\displaywidth{\hskip\@mathmargin\unhbox\z@\unskip}%
2553 \else % M+B > displaywidth
2554 \emdf@Ua
2555 \fi
2556 }
```

Some notation: M `\@mathmargin`, B the width of the equation body, I `\displayindent`, D `\displaywidth`, N the width of the equation number (aka the tag), S `\mintagsep`, C `\columnwidth`. If $M + B > displaywidth$, and if we assume M contains shrink, then the only solution left is to encroach into the `displayindent` space.

```

2557 \def\emdf@Ua{%
2558 \hbox to\columnwidth{%
2559 \ifdim\displayindent>\z@
2560 \hskip\displayindent minus\displayindent
2561 \fi
2562 \hskip\@mathmargin \unhbox\z@ \unskip
2563 }%
2564 \displayindent\z@ \displaywidth\columnwidth
2565 }
```

Find out first if the tag fits in ideal position. If so we can just plunk down box 2. Otherwise we need to do something more complicated.

```

2566 \def\emdf@R{%
2567 \setbox\tw@\hbox to\displaywidth{%
2568 \hskip\@mathmargin \unhcopy\z@\unskip\hfil\hskip\mintagsep\copy4
2569 }%
2570 \restore@hfuzz
2571 \ifnum\badness<\inf@bad \box\tw@ \else \emdf@Ra \fi
2572 }
```

We shift the equation number to line 2 if it does not fit within `\displaywidth`. Note that we do not first attempt to let the equation body shift leftward into the `\displayindent` space. If that is desired it will have to be done by hand by adding negative space at the beginning of the equation body. I don't expect this to arise very often in practice since most of the time `\displayindent` is zero anyway.

```

2573 \def\emdf@Ra{%
2574 \skip@\displayindent minus\displayindent
```

```

2575 \displayindent\z@ \displaywidth\columnwidth
2576 \spread@equation \everycr{}\tabskip\z@skip
2577 \halign{\hbox to\displaywidth{##}\cr
2578   \relax
2579   \ifdim\skip@>\z@ \hskip\skip@ \fi
2580   \hskip\@mathmargin\unhbox\z@\unskip\hfil\cr
2581   \noalign{\raise@tag}%
2582   \hfil\box4 \cr}%
2583 }

```

Find out first if the tag fits in ideal position. If so we can just plunk down box 2. Otherwise we need to do something more complicated.

```
2584 \def\emdf@L{%
```

Calculate the difference between M and $N + S$. If the latter is greater, we don't want to add any extra glue between the number and the equation body. Otherwise the amount that we want to add is x minus x where $x = M - (N + S)$. I.e., the distribution of spaces across the line is $N, S, x \text{ minus } x, B, h \text{ fil}$.

```

2585 \@tempdima\@mathmargin
2586 \advance\@tempdima-\wd4 \advance\@tempdima-\mintagsep
2587 \skip@\@tempdima minus\@tempdima
2588 \setbox\tw\hbox to\displaywidth{%
2589   \copy4\hskip\mintagsep
2590   \ifdim\skip@>\z@ \hskip\skip@\fi
2591   \unhcopy\z@\unskip
2592 }%
2593 \restore@hfuzz
2594 \ifnum\badness<\inf@bad \box\tw@ \else \emdf@La \fi
2595 }

```

If the equation body and equation number will not fit on the same line, we put the number on line 1 and the body on line 2, with the body positioned as for an unnumbered equation.

```

2596 \def\emdf@La{%
2597   \spread@equation \everycr{}\tabskip\z@skip
2598   \halign{\hbox to\displaywidth{##}\cr
2599     \box4 \hfil \cr
2600     \noalign{\raise@tag}%
2601     \hskip\@mathmargin\unhbox\z@\unskip\hfil\cr}%
2602 }

```

If someone has `\[\]` nested inside a minipage environment nested inside a numbered equation, the `mathdisplay` variables that are global will get out of whack unless we take extra care. So we make a stack and push all the variables before entering `mathdisplay` and pop them afterwards. But we can save a little work by not doing this at the top level, only at inner levels.

```

2603 \newtoks\mathdisplay@stack
2604 \let\mathdisplay@push\@empty
2605 \def\mathdisplay@push{%
2606   \begingroup
2607   \toks@\exp{\df@label}\@temptokena\exp{\df@tag}%
2608   \toks8\exp{\alt@tag}%
2609   \edef\@tempa{%
2610     \global\if@eqnsw\@nx\@eqnswtrue\else\@nx\@eqnswfalse\fi
2611     \global\iftag@\@nx\tag@false\else\@nx\tag@true\fi
2612     \gdef\@nx\df@label{\the\toks@}\gdef\@nx\df@tag{\the\@temptokena}%
2613     \gdef\@nx\alt@tag{\the\toks8}%
2614     \global\mathdisplay@stack{\the\mathdisplay@stack}%
2615   }%
2616   \global\mathdisplay@stack\exp{\@tempa}

```

```

2617 \endgroup
2618 }

2619 \let\mathdisplay@pop\@empty
2620 \def\mathdisplay@@pop{\the\mathdisplay@stack}

2621 \renewenvironment{equation}{%
2622   \incr@eqnum
2623   \mathdisplay@push
2624   \st@rredfalse \global\@eqnswtrue
2625   \mathdisplay{equation}%
2626 }{%
2627   \endmathdisplay{equation}%
2628   \mathdisplay@pop
2629   \ignorespacesafterend
2630 }

2631 \newenvironment{equation*}{%
2632   \mathdisplay@push
2633   \st@rredtrue \global\@eqnswfalse
2634   \mathdisplay{equation*}%
2635 }{%
2636   \endmathdisplay{equation*}%
2637   \mathdisplay@pop
2638   \ignorespacesafterend
2639 }

```

Note: \LaTeX defines the `displaymath` environment in terms of `\[` and `\]`.

```

2640 \DeclareRobustCommand{\[]{\begin{equation*}}
2641 \DeclareRobustCommand{\}]{\end{equation*}}

```

The usual `\endinput` to ensure that random garbage at the end of the file doesn't get copied by `docstrip`.

```
2642 \endinput
```

17 Credits

Much of the code for the `amsmath` package had its origin in `amstex.tex`, written by Michael Spivak. The initial work of porting `amstex.tex` to `amstex.sty` was done in 1988–1989 by Frank Mittelbach and Rainer Schöpf. In 1994 David M. Jones added the support for the `fleqn` option and did extensive improvements to the `align[at]` family of environments and to the equation number handling in general. Michael Downes at the AMS served as coordinator for the efforts of Mittelbach, Schöpf, and Jones, and has contributed various bug fixes and additional refinements over time.

Versions 1.0 and 1.1 of the package carried the name `amstex` instead of `amsmath`, to indicate its origins; the name was changed in 1994 to make it user-oriented rather than history-oriented.

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols			
\ast	10, 16	426, 440, 441,
\prime	10, 16, 296	442, 684, 776,
\prime	10, 16	782, 1163, 1227
\langle	10, 16, 38, 113, 114,	
\langle	351, 352, 353,	\neg 10, 16, 708, 711
\rangle	354, 420, 422,	\setminus 10, 16

- `.toc` 7
`\:` 117, 118
`\;` 120, 121, 274, 275, 276, 459
`\<` 11, 17
`\=` 11, 17, 709, 712
`\>` 11, 17
`\@@above` 158
`\@@abovewithdelims` 161
`\@@atop` 157
`\@@atopwithdelims` 160
`\@@eqno` 2469, 2470, 2473
`\@@italiccorr` 934
`\@leqno` 2468, 2471, 2474
`\@cover` 6, 156, 177
`\@coverline` .. 271, 272
`\@coverwithdelims` 159
`\@sqrt` 8, 251
`\@undefined` . 134, 149
`\@addtoreset` 861
`\amsmath@err`
 101, 225, 226,
 577, 675, 800,
 885, 890, 908,
 928, 952, 1194,
 1455, 1555, 2072
`\arrayboxrestore` 958
`\arrayparboxrestore`
 957, 1000
`\@backslashchar` .. 164
`\@badmath` .. 2498, 2517
`\@car` 105
`\@cdots` ... 11, 387,
 388, 389, 432, 467
`\@centering`
 . 2, 40, 1276, 2276
`\@checkend` 1145
`\@currentlabel`
 30, 945, 947
`\@currenvir`
 ... 794, 1131,
 1134, 1135,
 1160, 1240,
 1443, 2239, 2500
`\@display@init`
 35, 66, 1061, 2261
`\@displaytrue` 463
`\@eha` 225,
 226, 885, 908, 953
`\@ehd` 676
`\@elt` 1095
`\@empty` 20
`\@emptytoks` .. 106,
 278, 1132, 2512
`\@envbody` 36,
 37, 1127, 1128,
 1130, 1131, 1132
`\@eqnum` .. 30, 936, 939
`\@eqnswfalse` . 994,
 2522, 2610, 2633
`\@eqnswtrue`
 .. 1244, 1322,
 1523, 1542,
 1571, 2085,
 2246, 2610, 2624
`\@eqpen` 27, 35, 50, 67,
 68, 897, 898, 1072
`\@fleqnfalse` .. 55, 57
`\@fleqntrue` . 43, 52, 53
`\@genfrac` .. 7, 185, 189
`\@getpen` . 27, 900, 2531
`\@gobblethree` . 583, 589
`\@gtempa` 1097,
 1102, 1109, 1116
`\@highpenalty` 27
`\@iden` 2361
`\@ifnotempty` . 756, 758
`\@ifpackageloaded` . 90
`\@ifpackagewith` ... 55
`\@ignoretrue` 845
`\@ldots` 10, 343, 351,
 420, 422, 436,
 440, 441, 442, 821
`\@let@token` 65
`\@lign` .. 1539, 1562,
 2048, 2052, 2367
`\@lowpenalty` 27
`\@mathmargin` ... 2,
 2, 3, 3, 3, 3,
 39, 40, 44, 45,
 47, 48, 48, 50,
 51, 53, 53, 53,
 67, 72, 72,
 72, 73, 1307,
 1330, 1334,
 1338, 1633,
 1653, 1665,
 1682, 1705,
 1726, 1741,
 2268, 2271,
 2286, 2287,
 2308, 2399,
 2537, 2552,
 2562, 2568,
 2580, 2585, 2601
`\@mathmeasure` . 106,
 524, 693, 696, 697
`\@mathstyle` .. 185, 190
`\@medpenalty` 27
`\@nocnterr` 860
`\@nocounterr` 859
`\@saveprimitive` ...
 156, 157, 158,
 159, 160, 161,
 271, 2468, 2469
`\@spread@equation` .
 1002, 1060
`\@sptoken` . 234, 240,
 305, 310, 326, 334
`\@temp` 4, 7, 9, 14
`\@tempa` 60, 60, 60
`\@tempcnta`
 50, 54, 59, 60, 60
`\@tempcntb` . 50, 54,
 54, 58, 58, 60, 60
`\@tempd` 555
`\@tempdima` 53,
 55, 58, 58, 60,
 60, 1726, 1727,
 1749, 1789,
 1790, 1791,
 1795, 1802,
 1806, 1854,
 1855, 1862,
 1868, 1873,
 1881, 1935,
 1944, 1945,
 1946, 1953,
 1964, 1975,
 1997, 1999,
 2585, 2586, 2587
`\@tempdimb` 54,
 58, 58, 59,
 60, 60, 1773,
 1838, 1904,
 1907, 1910,
 1911, 1914,
 1936, 1945,
 1961, 1987,
 2001, 2002, 2010
`\@tempdimc` 54,
 1774, 1790,
 1839, 1854,
 1904, 1906, 1909
`\@testopt` 39, 1208, 1217
`\@totalleftmargin` .
 28, 28, 916
`\[` 3, 11, 17, 70, 75, 2640
`\]` ... 23, 23, 26, 26,
 27, 28, 28, 28,
 30, 32, 38, 291,
 319, 332, 905, 911
`\|*` 23
`\}` 394
`\]` 3, 11, 17, 70, 75, 2641
`\^` 11, 18
`_` 11, 18
`\`` 5
`_` 356
- A**
- `\above` 158, 173
`\abovedisplayskip` 919
`\abovewithdelims` ..
 161, 176

- `\acc@check` 1154, 1169, 1172, 1187, 1537, 1602, 1648, 1666, 1667, 1672, 1680, 1682, 1684, 1686, 1687, 1692, 1702, 1742, 1743, 1744, 1745, 1746, 1800, 1812, 1814, 1869, 1879, 1882, 1892, 1893, 1958, 1973, 1976, 1979, 1980, 1988, 2056
 - `\acc@error` 574, 575, 576
 - `\accentclass@` .. 15, 531, 532, 595, 621
 - accents package ... 18
 - `\Acute` 668
 - `\acute` 559, 668
 - `\add@amp` 1017
 - `\add@amps` . 63, 1017, 1535, 1617, 2156
 - `\addto@envbody` 1128, 1146, 1148
 - `\addto@hook` 510, 517, 527
 - `\addtocounter` 991
 - `\aftergroup` 2173, 2207
 - align environment .. . 31, 32, 32, 34, 36, 39, 44, 44, 44, 44, 44, 45, 48, 49, 51, 52, 52, 60, 62, 63, 63, 63, 67, 1461
 - align* environment 45, 1461
 - `\align@` 44, 1436, 1448, 1516
 - `\align@lengths` . 50, 50
 - `\align@preamble` 1528, 1613, 2045
 - `\align@recover` 1440, 1453
 - align[at] environ-ment 75
 - alignat environment 44, 44, 47, 47, 48, 49, 50, 50, 50, 1461
 - alignat* environment 1461
 - aligned environment 38, 38, 39, 39, 39, 39, 45, 65, 65, 1207
 - `\aligned@a` . 1209, 1215
 - alignedat environ-ment ... 38, 38, 38, 39, 39, 39, 39, 39, 39, 1216
 - `\alignedat@a` 1218, 1222
 - `\alignsafe@testopt` . 1201, 1208, 1217
 - `\alignsep@` 49, 50, 50, 50, 50, 50, 50, 51, 53, 53, 53, 54, 54, 54, 55, 55, 55, 55, 55, 58, 59, 59, 59, 59, 59, 1029, 1154, 1169, 1172, 1187, 1537, 1602, 1648, 1666, 1667, 1672, 1680, 1682, 1684, 1686, 1687, 1692, 1702, 1742, 1743, 1744, 1745, 1746, 1800, 1812, 1814, 1869, 1879, 1882, 1892, 1893, 1958, 1973, 1976, 1979, 1980, 1988, 2056
 - `\allowbreak` .. 680, 683
 - `\allowdisplaybreaks` 27, 869
 - `\alph` 852
 - `\alt@tag` 2479, 2487, 2521, 2527, 2608, 2613
 - `\AmS` 4, 102
 - `\ams@DeclareRobustCommand` 94, 99, 432
 - `\ams@def` 93, 98, 387, 714, 715
 - `\ams@newcommand` 90, 472, 473, 474, 535, 538, 734, 739, 741, 743
 - `\ams@renewcommand` . . 92, 97, 730, 732
 - amsbsy package ... 4, 4
 - `\AmSfont` ... 4, 102, 104
 - amsfonts package 1, 2, 2, 3, 3, 16, 16
 - `amslatex.tex` 1
 - AMSMATH package 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74
 - amsmath package 1, 1, 1, 2, 3, 3, 3, 4, 4, 4, 16, 16, 16, 24, 30, 37, 45, 65, 75, 75
 - `amsmath.sty` 3
 - amsopn package 4
 - amstex package 1, 1, 1, 3, 67, 75
 - `amstex.doc` 69
 - `amstex.sty` 75
 - `amstex.tex` 1, 32, 75, 75
 - amstext package 4
 - `\And` 276
 - `\andhelp@` 34, 1025, 1195, 1556
 - `\arabic` 858
 - array environment 23, 24, 24, 24, 25, 26
 - `\array` 24, 791, 841
 - `\arraycolsep` 24, 787, 789
 - `\arraystretch` 840
 - `\arrowfill@` 716, 722, 723, 724, 725, 726, 727
 - `\AtBeginDocument` .. . 46, 518, 710
 - `\AtEndOfPackage` 8
 - `\atop` 6, 23, 23, 157, 172
 - `\atopwithdelims` 160, 175
- ## B
- `\badness` 71, 2538, 2571, 2594
 - `\Bar` 673
 - `\bar` 564, 673
 - `\bBigg` 14
 - `\bBigg@` 519, 520, 521, 522, 523
 - `\begin` ... 28, 37, 37, 37
 - `\begin/endgroup` ... 68
 - `\begin@stack` 1132, 1142, 1143
 - `\begingroup` 28
 - `\belowdisplayskip` 913
 - `\bgroup` 246, 495, 764, 770, 776, 779, 1164, 1178, 1228, 1231, 1271, 1436, 1527, 2108, 2116, 2119, 2273, 2510
 - `\Big` 15, 519
 - `\big` 15, 15, 519
 - `\big@size` 525, 527
 - `\bigcap` 207, 208
 - `\bigcap@` 207, 208
 - `\bigcup` 209, 210
 - `\bigcup@` 209, 210
 - `\Bigg` 519
 - `\bigg` 519
 - `\Biggr` 405
 - `\biggr` 403
 - `\bigl` 15, 15
 - `\bigodot` 219, 220

- \backslash bigodot@ 219, 220
 \backslash bigoplus 217, 218
 \backslash bigoplus@ 217, 218
 \backslash bigotimes 215, 216
 \backslash bigotimes@ 215, 216
 \backslash Bigr 404
 \backslash bigr 15, 402
 \backslash bigsqcup 221, 222
 \backslash bigsqcup@ 221, 222
 \backslash biguplus 205, 206
 \backslash biguplus@ 205, 206
 \backslash bigvee 201, 202
 \backslash bigvee@ 201, 202
 \backslash bigwedge 203, 204
 \backslash bigwedge@ 203, 204
 \backslash binom 7, 7, 180
 \backslash binrel@ 21, 688, 690
 \backslash binrel@@ 21, 689, 691
 \backslash bl@ck 52
 \backslash black@ 1081,
 1252, 1489, 2111
bm package 4
 \backslash bmod 20, 677
 \backslash bold@false 386
 \backslash bold@true 385
 \backslash boldsymbol 4, 359, 386
 \backslash boldsymboldots@ . .
 11, 359, 384
 \backslash boxed 9, 273
 \backslash boxz@ 64,
 257, 1377, 1383,
 1398, 1402,
 1414, 2021,
 2026, 2038,
 2041, 2069,
 2144, 2163, 2166
 \backslash Breve 672
 \backslash breve 563, 672
 \backslash buildrel 13
- ### C
- \backslash c@MatrixCols 24
 \backslash c@MaxMatrixCols . .
 791, 792
 \backslash cal 16
 \backslash calc@shift@align .
 52,
 54, 1700, 1722,
 1756, 1821, 1918
 \backslash calc@shift@gather .
 33, 42, 1274, 1326
cases environment . .
 25, 832
 \backslash cases 25, 833
 \backslash cdotp 387, 468, 469, 470
 \backslash cdots 11,
 12, 432, 433, 434
 \backslash cfrac 685
 \backslash Check 666
 \backslash check 557, 666
 \backslash check@mathfonts . . 15
 \backslash checkat@false . . . 1432
 \backslash checkat@true 1426
 \backslash cl@ckpt 1098
 \backslash classnum@ 299, 300, 301
 \backslash cleaders 719
cmex 2
cmex10 4
cmex10 option 2, 3
 \backslash cmex@opt 3, 33,
 34, 36, 70, 77, 84
 \backslash collect@@body
 37, 37,
 37, 37, 1134, 1141
 \backslash collect@body 37,
 37, 1129, 1247,
 1451, 2087,
 2089, 2211, 2250
 \backslash colon 9, 283
 \backslash column@ 33,
 35, 39, 39, 54,
 54, 60, 1012,
 1014, 1020,
 1078, 1122,
 1153, 1177,
 1191, 1198,
 1291, 1551,
 1559, 1581,
 1609, 1616,
 1772, 1777,
 1781, 1782,
 1783, 1785,
 1786, 1837,
 1842, 1846,
 1847, 1848,
 1850, 1851,
 1905, 1906,
 1910, 1914,
 1934, 1939,
 2000, 2001,
 2004, 2010, 2149
 \backslash column@plus
 33, 1013,
 1179, 1184, 2059
 \backslash columnwidth 28, 73
 \backslash coprod 7, 197, 199, 200
 \backslash coprod@ 199, 200
 \backslash copy 256, 501,
 512, 516, 696,
 752, 2568, 2589
 \backslash count@ 58, 59, 60, 60
 \backslash cr 23, 28,
 906, 1156, 1199,
 1324, 1544,
 1572, 1617,
 2360, 2577,
 2580, 2582,
 2598, 2599, 2601
 \backslash crcl 63, 67, 68,
 728, 729, 737,
 738, 771, 773,
 780, 782, 1188,
 1211, 1233,
 1280, 1300,
 1528, 1613,
 2091, 2125,
 2154, 2168,
 2284, 2377, 2379
 \backslash ctagsplit@false . . 31
 \backslash ctagsplit@true . . . 30
- ### D
- \backslash dag 9
 \backslash dbinom 7, 181
 \backslash ddddot 538
 \backslash dddot 535
 \backslash Ddot 671
 \backslash ddot 562, 671
 \backslash DeclareFontEncoding
 533, 534
 \backslash DeclareFontShape . . 71
 \backslash DeclareMathAccent
 16, 18, 567
 \backslash DeclareMathDelimiter
 136, 138, 150, 152
 \backslash DeclareMathOperator
 4
 \backslash DeclareMathSymbol
 259,
 260, 261, 262,
 263, 264, 265,
 266, 267, 268, 269
 \backslash DeclareOption
 19, 20, 21, 22,
 23, 24, 28, 29,
 30, 31, 32, 42, 52
 \backslash DeclareRobustCommand
 5, 5, 99,
 111, 162, 177,
 180, 183, 272,
 344, 347, 435,
 439, 2640, 2641
 \backslash default@tag 23, 29,
 765, 777, 929,
 1001, 1175, 2118
 \backslash delayed@ 385, 419
 \backslash delimiter 125,
 130, 132, 145, 147
 \backslash df@label 30,
 31, 948, 966,
 968, 969, 2503,
 2520, 2528,
 2542, 2607, 2612
 \backslash df@tag 30,
 30, 31, 31, 942,
 962, 1567, 2100,
 2102, 2365,

- 2382, 2390, 2396, 2399, 2402
 2503, 2520, $\backslash display$.. 35, 66, 1061
 2525, 2527, $\backslash display@$
 2539, 2541, 35, 35, 39, 66,
 2546, 2607, 2612 1061, 1265, 1518
 $\backslash dfrac$ 6, 6, 178 $\backslash displaybreak$ 26, 27,
 $\backslash dft@tag$ 29, 929 50, 67, 68, 71, 875
 $\backslash dimen@$. 51, 59, 59, $\backslash displayindent$
 59, 59, 59, 253, 73, 73,
 255, 256, 648, 73, 2559, 2560,
 649, 662, 663, 2564, 2574, 2575
 698, 699, 1084, $\backslash displaylimits$.. 7,
 1089, 1327, 7, 19, 21, 488, 489
 1332, 1334, $displaymath$ environ-
 1338, 1339, ment . 70, 70, 75
 1340, 1349, $\backslash displaystyle$. 192,
 1350, 1351, 249, 273, 685,
 1352, 1387, 693, 696, 697,
 1388, 1389, 748, 1182,
 1645, 1646, 1185, 1232,
 1656, 1666, 1273, 1294,
 1670, 1680, 2048, 2052,
 1713, 1714, 2122, 2123,
 1792, 1793, 2281, 2367,
 1794, 1795, 2439, 2440,
 1796, 1800, 2458, 2459, 2512
 1801, 1802, $\backslash displaywidth$.. 35,
 1803, 1804, 42, 50, 52, 55,
 1805, 1806, 55, 58, 58, 73,
 1808, 1810, 73, 937, 1083,
 1811, 1814, 1271, 1310,
 1856, 1857, 1313, 1340,
 1858, 1860, 1346, 1352,
 1862, 1863, 1363, 1386,
 1867, 1868, 1388, 1645,
 1869, 1870, 1690, 1711,
 1871, 1872, 1715, 1718,
 1873, 1875, 1742, 1796,
 1877, 1878, 1804, 1805,
 1885, 1886, 1863, 1871,
 1889, 1892, 1872, 1880,
 1893, 1947, 1954, 1963,
 1948, 1949, 1974, 2269,
 1951, 1953, 2402, 2510,
 1954, 1958, 2552, 2564,
 1959, 1960, 2567, 2575,
 1961, 1962, 2577, 2588, 2598
 1963, 1964, $\backslash displaywidth@$
 1966, 1969, . 1716, 1718, 2028
 1970, 1972, $\backslash divide$ 300,
 1974, 1975, 1360, 1657,
 1976, 1977, 1661, 1667,
 1979, 1980, 1671, 1676,
 1984, 1986, 1684, 1693,
 1987, 1988, 1744, 1782,
 1989, 1990, 1810, 1847,
 2308, 2309, 1877, 1883,
 2310, 2312, 1969, 1977, 2006
 2394, 2395, $\backslash DN@$ 230, 234,
 235, 241, 242,
 279, 306, 307,
 311, 312, 320,
 327, 330, 335,
 336, 351, 422,
 426, 432, 436,
 440, 976, 1240,
 1436, 1440,
 1443, 1448, 2239
 $docstrip$ 75
 $\backslash Dot$ 670
 $\backslash dot$ 561, 670
 $\backslash doteq$ 13, 460
 $\backslash dots$ 11, 11, 347
 $\backslash DOTs@$ 320,
 326, 330, 373, 412
 $\backslash DOTs@@$ 321, 325
 $\backslash DOTsB$ 7,
 200, 202, 204,
 206, 208, 210,
 212, 214, 216,
 218, 220, 222,
 274, 275, 276,
 289, 446, 448,
 450, 452, 453,
 454, 455, 456,
 457, 458, 459, 461
 $\backslash dotsb$ 433
 $\backslash dotsb@$... 301, 302,
 361, 363, 372,
 375, 379, 381, 389
 $\backslash dotsc$ 360, 439
 $\backslash DOTSCASE@$ 318, 328,
 329, 330, 375, 414
 $\backslash DOTSI$ 288, 464, 465,
 472, 473, 474, 475
 $\backslash dotsi$ 376, 388
 $\backslash dotsm$ 434
 $\backslash dotso$ 435
 $\backslash dotso@$ 358, 417
 $\backslash dotsspace@$
 824, 827, 828, 830
 $\backslash DOTSX$ 290
 $\backslash dp$ 253, 508, 529, 652,
 694, 702, 1087,
 1396, 1411,
 2036, 2065,
 2066, 2340, 2375
 $\backslash dspbrk@$ 875
 $\backslash dspbrk@context$ 875,
 1174, 1229,
 1264, 1517,
 2117, 2262, 2501
 $\backslash dspbrk@lvl$ 26, 28,
 35, 866, 881,
 899, 900, 1072,
 1078, 2261,
 2530, 2531, 2532
 $\backslash dt@pfalse$ 1068

- `\dt@ptrue` 1063
E
`\edef` 26
`\egroup` 72, 250, 497,
773, 782, 1211,
1213, 1252,
1490, 1493,
2092, 2093,
2111, 2112,
2328, 2353, 2537
`\emdf@L` 2584
`\emdf@La` ... 2594, 2596
`\emdf@R` 2566
`\emdf@Ra` ... 2571, 2573
`\emdf@U` 2549
`\emdf@Ua` ... 2554, 2557
`\end` 37, 37, 37, 37
`\endalign` ... 1464,
1469, 1474,
1479, 1484,
1504, 1509, 1514
`\endaligned` 1220, 1235
`\endarray` . 24, 787, 835
`\endgather` 1259
`\endinput` 75
`\endmathdisplay` ...
. 2516, 2627, 2636
`\endmathdisplay@a` .
. 2110, 2518, 2524
`\endmathdisplay@fleqn`
..... 2526, 2536
`\endmatrix` ... 811,
813, 817, 819, 820
`\endmultline` 2258
`\endmultiline@math` .
. 2281, 2323, 2333
`\endsplit` 2087
`\endsubarray` 775
`\env@cases` ... 833, 837
`\env@matrix`
785, 789, 809,
813, 815, 819, 820
environments:
 `align` 31, 32, 32,
34, 36, 39, 44,
44, 44, 44, 44,
45, 48, 49, 51,
52, 52, 60, 62,
63, 63, 63, 67, 1461
 `align*` 45, 1461
 `align[at]` 75
 `alignat` 44,
44, 47, 47, 48,
49, 50, 50, 50, 1461
 `alignat*` 1461
 `aligned` 38,
38, 39, 39, 39,
39, 45, 65, 65, 1207
 `alignedat`
... 38, 38, 38,
39, 39, 39, 39, 1216
 `array` 23,
24, 24, 24, 25, 26
 `cases` 25, 832
 `displaymath`
..... 70, 70, 75
 `eqnarray` ... 26, 30
 `equation` ... 70, 70
 `equation*` 70
 `flalign` . 44, 48,
49, 51, 52, 59, 1461
 `flalign*` 1461
 `gather`
32, 33, 34, 36,
38, 40, 62, 63, 1249
 `gather*` 1249
 `gathered` .. 40, 1223
 `matrix` .. 24, 24, 784
 `multiline` 33, 35,
65, 66, 69, 69, 2252
 `multiline*` 2252
 `pmatrix` 24
 `Sb` 23
 `smallmatrix` . 24, 776
 `Sp` 23
 `split` 42,
61, 61, 62, 63,
63, 63, 63, 63, 2080
 `subarray`
23, 23, 23, 23, 763
 `xalignat`
.. 44, 44, 51, 1461
 `xalignat*` 1461
 `xxalignat` .. 44,
44, 48, 49, 51, 1461
`eqnarray environment`
..... 26, 30
`\eqnarray` 3
`\eqno` 72, 2469, 2470, 2471
`\eqnshift@` . 34, 49,
50, 52, 53, 53,
54, 55, 58, 59,
59, 59, 59, 59,
59, 1028, 1330,
1346, 1347,
1355, 1356,
1360, 1362,
1363, 1364,
1366, 1367,
1413, 1526,
1601, 1653,
1656, 1657,
1665, 1670,
1671, 1672,
1679, 1688,
1690, 1691,
1692, 1693,
1697, 1698,
1708, 1740,
1741, 1789,
1867, 1880,
1881, 1882,
1883, 1885,
1887, 1889,
1960, 1989, 1990
`\eqref` 26, 865
equation environment
..... 70, 70
`\equation` 3, 30, 44
equation* environ-
ment 70
`\every@math@size` 14,
15, 510, 517, 527
`\everycr` 35,
66, 1064, 1290,
1608, 2576, 2597
`\everydisplay` 463
`\everymath` 2512
`\ex@` . 536, 539, 738, 778
`\ExecuteOptions` ... 53
`\expandafter` 37
`\ext@arrow`
745, 746, 761, 762
`\extra@` 407,
420, 422, 429, 442
`\extrap@` . 425, 432, 436
F
`\f@series` 4, 4, 105
`\fbox` 9, 273
`\fi` 28, 28
`\field@lengths`
..... 1574, 1604
`\fieldlengths@`
..... 54, 1574,
1775, 1840, 1937
`\finsm@sh` 703
flalign environment
..... 44, 48,
49, 51, 52, 59, 1461
flalign* environment
..... 1461
fleqn option 2, 3, 42,
48, 48, 48, 50,
50, 51, 51, 51,
51, 65, 67, 69, 75
`fleqn.clo` 3, 3, 3
`\FN@` . 233, 234, 235,
239, 241, 242,
246, 278, 281,
307, 312, 327,
336, 356, 357,
422, 432, 438, 444
`\fontdimen`
766, 767, 768, 1031
`\fontencoding` 79

- `\fontfamily` 79
`\frac` 6, 7, 165, 177, 685
`\frozen@everymath` .
 106, 590, 610
- G**
- `gather` environment .
 32, 33, 34, 36,
 38, 40, 62, 63, 1249
`gather*` environment 1249
`\gather@` 1245, 1261, 2109
`\gather@split` 2089, 2097
`gathered` environment
 40, 1223
`\gdisplaywidth@` ...
 1279,
 1311, 1313, 1373
`\genfrac` 6, 6,
 165, 178, 179,
 180, 181, 182, 183
`\getdsp@pen` .. 870, 872
`\getmathch@` .. 300, 369
`\glb@settings` 15
`\global` 9, 25
`\gmeasure@` . 1267, 1283
`\gobble@tag`
 ... 927, 928, 2267
`\Grave` 669
`\grave` 560, 669
`\gtest@false` 9, 287,
 293, 297, 306,
 311, 314, 320,
 335, 342, 405, 414
`\gtest@true`
 .. 9, 286, 294,
 298, 305, 310,
 315, 326, 334,
 338, 390, 409, 414
- H**
- `\halign` 13, 23,
 48, 48, 50, 50,
 50, 58, 1271,
 1293, 1527,
 1613, 2273,
 2366, 2577, 2598
`\Hat` 20, 665
`\hat` .. 17, 20, 556, 665
`\hbox` 50, 103,
 106, 247, 254,
 500, 503, 514,
 537, 540, 587,
 646, 647, 650,
 651, 699, 719,
 748, 750, 751,
 754, 828, 933,
 936, 1088, 1638,
 2134, 2143,
 2147, 2169,
- 2182, 2191,
 2196, 2274,
 2298, 2307,
 2342, 2440,
 2459, 2510,
 2541, 2552,
 2558, 2567,
 2577, 2588, 2598
`\hdots` 821
`\hdots@for` 823, 825, 826
`\hdotsfor` 822
`\hexnumber@` 543
`\hfil` 66, 66, 67, 729,
 737, 770, 771,
 779, 780, 1180,
 1186, 1232,
 1415, 2046,
 2055, 2120,
 2124, 2278,
 2282, 2537,
 2568, 2580,
 2582, 2599, 2601
`\hfill` 686, 719,
 829, 1088, 2124
`\hfilneg` 66,
 2289, 2325,
 2351, 2414,
 2420, 2441, 2460
`\hfuzz` 2508, 2509
`\hookleftarrow` ... 458
`\hookrightarrow` .. 457
`\hskip` 66
`\hspace` 5
`\ht` 253, 508,
 528, 652, 694,
 702, 1086, 1409,
 2061, 2062, 2371
- I**
- `\ialign` 728,
 737, 770, 779,
 1178, 1231, 2119
`\idotsint` 475
`\if` 5
`\if@display`
 . 13, 462, 681,
 683, 1439, 2081
`\if@eqnsw` 961,
 990, 1533, 1550,
 1563, 2383,
 2386, 2525, 2610
`\if@fleqn` 37,
 1306, 1329,
 1633, 1651,
 1664, 1681,
 1689, 1703,
 1722, 1756,
 1821, 1918,
 2216, 2268,
- 2270, 2275,
 2285, 2398,
 2434, 2507, 2526
`\ifbold@` 416, 418
`\ifcase` 47
`\ifcheckat@` 1417, 1552
`\ifctagsplit@`
 26, 2141, 2162,
 2189, 2201, 2221
`\ifdt@p` 1067
`\iff` 459
`\ifgtest@` 9,
 285, 286, 287,
 363, 367, 369,
 371, 372, 374,
 378, 379, 381,
 408, 411, 413,
 420, 422, 430, 442
`\ifhbox` 1641
`\iffinalign@` ... 63,
 64, 998, 2128, 2177
`\ifinany@` 33
`\ifingather@` . 998,
 1434, 1491, 1518
`\ifinner` 13, 2082
`\ifmeasuring@` .. 31,
 33, 878, 964,
 1009, 2049, 2053
`\ifmmode` 5
`\ifshifttag@` .. 33,
 1010, 1374,
 1391, 2295,
 2334, 2422, 2443
`\ifst@rred`
 .. 1008, 1244,
 1319, 1322,
 1523, 1532,
 1542, 1549,
 1571, 2085, 2243
`\iftag@` ... 31, 962,
 975, 1007, 1538,
 1567, 2099,
 2293, 2332,
 2384, 2421,
 2442, 2482, 2611
`\iftagsleft@` .. 27,
 935, 985, 1279,
 1309, 1333,
 1361, 1372,
 1408, 1710,
 1722, 1756,
 1821, 1901,
 1918, 2016,
 2060, 2094,
 2223, 2255,
 2292, 2368,
 2397, 2411,
 2437, 2474, 2546
`\ifxxat@` ... 1417, 1520

- `\ignorespacesafterend` 25, 845, 856, 1253, 1499, 2256, 2629, 2638
`\iiiint` 474
`\iiint` 473
`\iint` 472
`\ilimits@` 19, 20, 464, 465, 482, 489, 497
`\impliedby` 275
`\implies` 274
`\inalign@false` ... 1001
`\inalign@true` ... 1517
`\incr@eqnum` 961, 991, 993, 997, 1005, 2622
`\inf@bad` ... 5, 109, 2538, 2571, 2594
`\ingather@false` .. 1001
`\ingather@true` ... 1262
`\insplit@` 1262, 1447, 2115
`\int` 464
`\intdots@` 467, 478
`\interdisplaylinepenalty` ... 26, 27, 35, 50, 868, 870, 899
`\intertext` 28, 28, 28, 908
`\intertext@` 28, 908, 1265, 1517
`\intkern@` 466, 478, 479, 480
`\intop` 464, 477, 479, 480, 481
`\ints@a` 484, 486
`\ints@b` 487, 488, 489, 493
`\ints@c` 477, 491
`\Invalid@@` 225, 226, 885, 908
`\invalid@tag` 29, 31, 928, 929, 976, 2483

J
`\joinrel` 446, 448, 450, 452, 453, 454, 457, 458
`\jot` 1059

K
`\kern` 103, 112, 461, 516, 536, 539, 654, 663, 687, 689, 691, 699, 738, 1154, 1373, 1389, 1413, 1537, 2017, 2028, 2030
`\keybin@` 338, 362

L
`\label` 30, 30, 30, 30, 30, 31, 31, 31, 31, 31, 31, 66, 66, 68
`\label@in@display` . 30, 71, 948, 1263, 1521, 2359, 2502
`\lastbox` .. 50, 1630, 1639, 2131, 2137, 2139, 2180, 2185, 2187
`\lbrace` 815, 839
`\ldots` 11, 344
`\left` . 525, 808, 813, 815, 819, 820, 839
`\Leftarrow` 452, 454, 725, 727
`\leftarrow` ... 450, 453, 458, 722, 724
`\Leftarrowfill@` .. 725
`\leftarrowfill@` ... 722, 733, 742, 762
`\leftmargini` ... 3, 48
`\Leftrightarrowfill@` 727
`\leftrightarrowfill@` 724, 735, 744
`\leftroot` 225, 231, 236, 238, 239
`\leftroot@` 228, 238, 239, 246, 256, 257
`\lendmultline@` 66, 2255, 2324
`\lendsplit@` 2094, 2175
`\leqno` 72, 2468, 2470, 2471
`\let` 11, 37, 37
`\Let@` ... 23, 28, 765, 777, 905, 1174, 1229, 1265, 1517, 2117, 2260
`\lhook` 457
`\limits` 13, 23, 461, 487, 536, 539, 689, 691, 755
`\lineht@` ... 34, 61, 63, 1035, 1378, 1396, 1409, 1411, 1543, 2022, 2036, 2061, 2062, 2065, 2066, 2155, 2302, 2340, 2371, 2375
`\lineskip` 24
`\lineskiplimit` 24, 769, 778, 1069

`\linewidth` 28, 28, 915, 916
`\llap` ... 1392, 1402, 2032, 2041, 2336
`\Longleftarrow` 275, 451
`\longleftarrow` ... 449
`\Longleftrightarrow` 454, 459
`\longlefttrightarrow` 453
`\longmapsto` 456
`\Longrightarrow` 274, 447
`\longrightarrow` 445, 456
`\loop` ... 1637, 1728, 1760, 1825, 1922
`\lower` 103, 515
`\lq` 67
`\ltx@label` 30, 948, 968, 2528, 2542
`\lVert` 145, 149, 150, 820
`\lvert` 130, 134, 136, 819

M
`\m@ne` . 626, 628, 643, 991, 1072, 1078, 1166, 1215, 1425, 1487, 1502, 1507, 1512, 1663, 1677, 1735, 1767, 1781, 1832, 1846, 1929, 2005, 2261, 2530, 2532
`\m@th` .. 30, 107, 248, 252, 254, 273, 351, 717, 729, 737, 749, 771, 779, 780, 827, 828, 933, 1182, 1185, 1232, 1273, 1294, 2048, 2052, 2122, 2123, 2281, 2367, 2439, 2440, 2458, 2459, 2512
`\macc@a` 622, 639
`\macc@adjust` 656, 657, 661
`\macc@code` 621, 654, 656
`\macc@depth` 582, 586, 593, 605, 643, 644
`\macc@group` 606, 610, 618, 619
`\macc@kerna` 649, 654, 662
`\macc@kernb` .. 653, 662
`\macc@nested` . 597, 607
`\macc@nested@a` 612, 613, 617

- \macc@nucleus 594, 600, 645
- \macc@palette 19, 616, 622, 642
- \macc@set@skewchar 19, 609, 611, 619, 624
- \macc@skewchar 635, 646, 647
- \macc@style 19, 641, 642
- \macc@test . . . 588, 605
- \macc@tmp 586, 592, 605
- \macro@ . . 314, 370, 411
- \macro@@ 316, 317
- \make@df@tag 30, 30, 32, 942, 980
- \make@df@tag@@ 942, 2490
- \make@df@tag@@@ 942, 2492
- \make@display@tag 31, 960, 1321, 1539, 2300, 2307, 2342, 2346
- \makesm@sh 705
- \maketag@@ . . 29, 30, 932
- \maketag@@@ 29, 30, 932, 945
- \mapsto 455
- \mapstochar . . 455, 456
- \math@ . . . 293, 366, 377
- \math@bgroup 609
- \math@cr . . . 27, 28, 50, 67, 68, 896, 905, 1252, 1489, 2111, 2327, 2352
- \math@cr@ 27, 28, 897, 901, 902
- \math@cr@@ 28, 902, 903
- \math@cr@@@ 23, 23, 28, 28, 38, 63, 903, 906, 1168, 1171, 1266, 1302, 1519, 1607, 1615, 2360
- \math@cr@@@align 1519, 1531
- \math@cr@@@align@measure 1546, 1607
- \math@cr@@@aligned 39, 1152, 1171, 1190
- \math@cr@@@alignedat 1168, 1190
- \math@cr@@@gather 41, 1266, 1318
- \math@egroup . . 19, 609
- \mathaccent 16, 17, 17, 17, 18, 18, 547, 595, 654, 656
- \mathaccentV 16, 16, 17, 17, 17, 18, 542, 584, 588, 612
- \mathalpha 567
- \mathbin 677
- \mathbin@ 304, 305, 306, 378
- \mathch@ 297, 368
- \mathchar 195, 276
- \mathchardef 504, 621, 635, 708, 709, 711, 712
- \mathchoice . . . 18, 19, 20, 23, 249, 466, 467, 494, 496
- \mathclose . . . 139, 153
- \mathcode 504, 708, 709, 711, 712
- \mathdisplay 2497, 2625, 2634
- \mathdisplay@pop 2506, 2620
- \mathdisplay@push 2505, 2605
- \mathdisplay@pop 2506, 2619, 2628, 2637
- \mathdisplay@push 2505, 2604, 2623, 2632
- \mathdisplay@stack 2603, 2614, 2616, 2620
- \mathellipsis 10, 343, 345
- \mathgroup 610, 618, 619, 626, 628, 629
- \mathindent . . 3, 3, 3, 45
- \mathinner 387
- \mathit 16
- \mathop 13, 21, 21, 461, 495, 536, 539, 689, 691, 699, 747
- \mathopen 137, 151
- \mathord 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269
- \mathpalette 616, 704, 714, 731, 733, 735, 740, 742, 744
- \mathpunct 283
- \mathrel 9, 461, 714, 715, 747
- \mathrel@ 309, 310, 311, 380
- \mathring 567, 569
- \mathsm@sh . . . 704, 714
- \Mathstrut@ 499
- \Mathstrutbox@ 499, 528, 529
- \mathsurround 610
- matrix environment 24, 24, 784
- \matrix 24, 24, 785
- \matrix@check . . 24, 785, 793, 809, 833
- \matrix@error . . 795, 799
- \maxcol@width 47, 1595, 1906, 1910, 1914, 2001, 2010
- \maxcolumn@widths 47, 47, 1594, 1596, 1634, 1642
- \maxfields@ . . . 49, 49, 1016, 1165, 1166, 1167, 1191, 1424, 1425, 1430, 1535, 1551, 1559, 1617, 1621, 1622, 1625, 1660, 1675
- \mb@b 702
- \mb@t 702
- \mb@tb 702
- \mdots@ 348, 357
- \mdots@@ . . 357, 358, 385
- \meaning 197, 364, 410, 506, 545
- \meaning@ 294, 298, 316, 364, 365, 366, 368, 369, 372, 373, 377, 378, 380, 410, 411, 412
- \meaning@@ . . . 365, 370
- \measure@ 49, 1524, 1598
- \measuring@false 1193, 1554, 2359
- \measuring@true 1285, 1600, 2357
- \medmuskip . . . 117, 119, 677, 679, 717
- \medspace 118
- \MessageBreak 164, 166, 578, 893, 1456, 2230
- \minalignsep . . 39, 48, 50, 50, 52, 54, 55, 59, 1032, 1172, 1649, 1686, 1687, 1713, 1745, 1746

- 1792, 1811,
 1812, 1856,
 1878, 1879,
 1947, 1972, 1973
`\mintagsep` . 34, 48,
 48, 49, 53, 55,
 73, 1031, 1327,
 1727, 1794,
 1808, 1858,
 1860, 1875,
 1949, 1951,
 1966, 1984,
 2568, 2586, 2589
`\mkern` 256, 257,
 284, 466, 468,
 469, 470, 494,
 496, 677, 679,
 681, 682, 683,
 684, 718, 719,
 720, 750, 751,
 756, 757, 758,
 759, 827, 828, 830
`\mmeasure@` 66, 2266, 2355
`\mod` 20, 683
`\moveleft` . . . 2144,
 2153, 2192, 2197
`\mskip` 112, 123, 254,
 283, 284, 677, 679
`\mspace` 123
`\multicolumn` 826
`\MultiIntegral` 472,
 473, 474, 475, 476
`\multilimits@` 24
`\multiply` . . . 1167,
 1349, 1430,
 1793, 1801,
 1857, 1948, 1959
 multiline environment
 33, 35,
 65, 66, 69, 69, 2252
`\multline*` environ-
 ment 2252
`\multline@` . 2248, 2259
`\multline@indent` . .
 . . 2287, 2290,
 2305, 2316, 2319
`\multlinegap` . . 65,
 65, 65, 67,
 2232, 2290,
 2326, 2335,
 2349, 2415,
 2423, 2429,
 2444, 2450, 2461
`\multlinetaggap` 65,
 65, 2232, 2309,
 2310, 2345,
 2396, 2426, 2447
`\multlinetaggap@` . . 65
- N
`\NeedsTeXFormat` 1
`\negmedspace` 119
`\negthickspace` . . . 122
`\negthinspace` 116
`\new@ifnextchar` 27,
 790, 838, 875, 902
`\newbox` 499, 511
`\newcount` . 109, 227,
 228, 299, 318,
 582, 792, 866,
 1011, 1012, 1016
`\newcounter` 843
`\newdimen` 530, 1028,
 1029, 1030,
 1033, 1034, 1035
`\newhelp` . 29, 922, 1025
`\newif` . 2, 9, 26, 27,
 37, 416, 462,
 998, 999, 1007,
 1008, 1009,
 1010, 1417, 1419
`\newmuskip` 824
`\newskip` 39, 2232, 2234
`\next@` . . 31, 32, 230,
 231, 232, 235,
 236, 237, 242,
 243, 244, 246,
 278, 281, 307,
 312, 321, 323,
 331, 337, 356,
 422, 431, 432,
 437, 438, 444,
 980, 982, 1153,
 1154, 1156,
 1245, 1247,
 1451, 2248, 2250
`\nextii@`
 230, 233, 305,
 307, 310, 312,
 326, 327, 334,
 336, 420, 422, 424
`\nextiii@` 231, 239
`\nextiv@` 233, 234
`\nextix@` 243, 245
`\nextv@` . . 234, 235, 236
`\nextvi@` 236, 238
`\nextvii@` 239, 240
`\nextviii@` 241, 242, 243
`\noaccents@` 532, 533, 534
`\noalign` . . . 28, 28,
 729, 738, 904,
 912, 1065, 1082,
 1290, 1608,
 2155, 2581, 2600
`\nobreakdash` . . 9, 277
`\nogood@displaybreak`
 875
`\noindent` . . . 918, 1088
- `\nointerlineskip` . .
 . . . 729, 738, 1085
`\nolimits` 7,
 13, 20, 22, 23, 697
`\nomath@env`
 . 1239, 1442, 2238
`\nonmatherr@` . 602,
 675, 1160, 1225
`\nonscript` 283, 677, 679
`\nonumber` 31, 32, 32,
 32, 979, 988,
 989, 1319, 1494,
 1532, 1549,
 2103, 2244, 2485
`\normalbaselines` . .
 . . . 914, 1376,
 1394, 2020,
 2034, 2297, 2338
`\normalcolor` . 936, 939
`\normalfont` . . 537,
 540, 933, 936, 939
`\normallineskiplimit`
 . . . 515, 516, 1070
`\not` 361
`\not@` . 333, 334, 335, 372
`\notag` 32, 988
`\null` 38,
 776, 1163, 1227,
 1395, 2035, 2339
`\nulldelimiterspace`
 524, 687
`\number`
 4, 6, 15, 32, 32,
 130, 132, 145,
 147, 301, 375,
 635, 1022, 1122
`\numberwithin` 858
- O
`\of` 247
`\oint` 465
`\ointop` 465
`\omit` 1536, 1547
`\openup` 26, 1059
`\operator@font`
 678, 682, 684
`\or` 47, 50, 193,
 301, 376, 873,
 882, 1042, 1043,
 1054, 1055,
 1108, 1578,
 1582, 1636,
 1642, 1659, 1674
`\over` . 6, 6, 13, 156, 171
`\overarrow@`
 728, 731, 733, 735
`\overfullrule` 35
`\overleftarrow` . . . 732
`\overleftrightarrow` 734

- $\overline{}$. . . 9, 271, 272
 $\overrightarrow{}$. . . 730
 $\overset{}{}$ 21, 688
 $\overwithdelims{}{}$ 159, 174
- P**
- $\p@equation$ 947
 \PackageError 101
 \PackageWarning 163, 2229
 \PackageWarningNoLine 550
 \par 3, 24, 37
 \parshape 916
 ∂ 19
 \PassOptionsToPackage 23, 25
 $\place@tag$ 60, 1540, 2015
 $\place@tag@gather$ 32, 33, 1278, 1371
`plain.tex` 11, 24, 25, 35
 $\plainroot@$ 232, 237, 238, 244, 245, 247
`pmatrix` environment 24
 \pmb 24, 809
 \pmb 4
 $\pmod{}$ 20, 682
 \pod 20, 680, 682
 \postdisplaypenalty 913, 2531
 \predisplaypenalty 919
 \prevdepth 1084, 1089
 \primfrac 162, 171, 172, 173, 174, 175, 176
 $\print@eqnum$ 961, 993, 996, 1004
 $\process@envbody$ 1131, 1135, 1136, 1150
 \ProcessOptions 3, 54
 \prod 211, 212
 $\prod@$ 211, 212
 \protect 5, 26
 $\protected@edef$ 849
 \ProvidesPackage 3
 $\push@begins$ 37, 1138, 1142
- Q**
- \quad 841
- R**
- $\r@t$ 249, 250, 252
 \raise 256
 $\raise@tag$ 32, 32, 35, 985, 987, 1079, 1291, 1379, 1393, 1609, 2023, 2033, 2265, 2303, 2337, 2581, 2600
 \raisetag 32, 32, 984
 \rangle 396
 \rbrace 395, 817
 \rceil 397
 \ref 865
 \refstepcounter 26, 848, 997, 1005
 \relax 5, 27, 28, 65
 $\relaxnext@$ 229, 304, 309, 325, 333, 350, 417
 \Relbar 448, 452, 715, 725, 726, 727
 \relbar 446, 450, 714, 722, 723, 724
 $\rendmultline@$ 67, 2255, 2331
 $\rendsplitleft@$ 2094, 2127
 \renewenvironment 24, 25
 \repeat 1643, 1736, 1768, 1833, 1930
 \RequirePackage 87, 88, 89
 $\reset@equation$ 1003, 2098, 2478, 2485
 $\reset@strutbox@$ 15, 35, 513, 517, 518, 1058
 $\resetMathstrut@$ 499
 $\restore@hfuzz$ 2496, 2508, 2511, 2550, 2570, 2593
 $\restore@math@cr$ 28, 765, 777, 906, 1229, 2117, 2263
 $\restorealignstate@$ 1106, 1492
 $\restorecolumn@$ 1120, 1212
 $\restorecounters@$ 1093, 1305, 1620, 2408
 \rfloor 398
 \rgroup 399
 ρ 458
 $\RIfM@$ 1159, 1224, 1238, 2237
 \right 401, 525, 811, 813, 817, 819, 820, 835
 \rightarrow 448, 454, 726, 727
 \rightarrowtail 446, 453, 455, 457, 723, 724
 $\rightarrowtail@$ 726
 $\rightarrowtail@$ 723, 731, 740, 745, 761
 $\rightdelim@$ 11, 390, 408
 \rmoustache 400
 \romannumeral 1022
 \root 229
 \rootbox 247, 256
 $\row@$ 42, 42, 63, 1011, 1111, 1270, 1323, 1328, 1525, 1534, 1548, 1611, 1729, 1730, 1735, 1749, 1761, 1762, 1767, 1775, 1788, 1826, 1827, 1832, 1840, 1853, 1923, 1924, 1929, 1937, 1943, 2018, 2031, 2152, 2261, 2363, 2369
 \rq 67
 \rVert 147, 152, 820
 \rvert 132, 138, 819
- S**
- $\savealignstate@$ 1106, 1436
 $\savecolumn@$ 1120, 1162
 $\savecounters@$ 1093, 1288, 1605, 2362
 $\savefieldlength@$ 1574, 2049, 2053
 $\saveshift@$ 1052, 1733, 1750, 1752, 1765, 1797, 1799, 1830, 1864, 1866, 1927, 1955, 1957
 $\savetaglength@$ 63, 1036, 1299, 1570, 2151
`Sb` environment 23
 \scriptfont 766, 767, 768
 \scriptscriptstyle 23, 193, 248, 250
 \scriptstyle 193, 250, 750, 751, 771, 779, 780
 $\select@group$ 589
 $\set@field$ 61, 2050, 2054, 2058

- `\set@fontsize` 15
 - `\set@gather@field` .
..... 1275, 1371
 - `\set@mathaccent` ...
..... 16, 541, 555
 - `\setb@ck` 30
 - `\setbox` 106,
247, 254, 500,
503, 514, 587,
646, 647, 650,
651, 694, 695,
748, 749, 1289,
1395, 1606,
1629, 1630,
1638, 1639,
2035, 2116,
2129, 2131,
2134, 2137,
2139, 2147,
2176, 2178,
2180, 2182,
2185, 2187,
2307, 2339,
2364, 2440,
2459, 2510,
2541, 2567, 2588
 - `\setboxz@h` ... 252,
280, 1273, 1277,
1294, 1298,
1539, 1562,
2048, 2052,
2150, 2367,
2385, 2439, 2458
 - `\setcounter` 850, 851, 855
 - `\setlength` 65
 - `\shd@ts@for` .. 823, 825
 - `\shift@tag`
. 1036, 2018, 2031
 - `\shifttag@false` ...
..... 1268,
1381, 1400, 2405
 - `\shifttag@true` 1335,
1341, 1353, 2403
 - `\shoveleft` 44,
69, 69, 2361, 2411
 - `\shoveright` ... 44,
69, 69, 2361, 2411
 - `showkeys` package ..
..... 71, 71, 71
 - `\sideset` 21, 692
 - `\sixt@n` 876
 - `\size@update` .. 15, 15
 - `\SK@@label` . 2476, 2488
 - `\SK@equationtrue` ..
..... 2476, 2502
 - `\SK@tagform@` 2487, 2488
 - `\skewchar` 628, 632
 - `\skip@` 984, 985, 2574,
2579, 2587, 2590
 - `\slimits@` ... 7, 21,
22, 200, 202,
204, 206, 208,
210, 212, 214,
216, 218, 220, 222
 - `smallmatrix` environ-
ment ... 24, 776
 - `\smash` 21, 701
 - `Sp` environment 23
 - `\span` .. 1279, 1528, 1613
 - `split` environment 42,
61, 61, 62, 63,
63, 63, 63, 63, 2080
 - `\split` 1262, 1447
 - `\split@` 2142,
2148, 2163,
2165, 2173,
2190, 2195,
2202, 2204, 2207
 - `\split@al@a` 2211, 2212
 - `\split@al@tagcheck`
..... 2216, 2220
 - `\split@aligned`
..... 2083, 2209
 - `\split@err` . 2071, 2087
 - `\split@tag` .. 2096,
2101, 2105, 2109
 - `\split@warning`
. 2213, 2228, 2511
 - `\spread@equation` ..
. 15, 33, 1002,
1058, 1063,
1176, 1230,
2107, 2576, 2597
 - `\sqrt@sign` 8, 8, 251, 252
 - `\st@rredfalse` .. 44,
1250, 1462,
1472, 1487,
1507, 2253, 2624
 - `\st@rredtrue` .. 44,
1257, 1467,
1477, 1482,
1502, 1512,
2258, 2485, 2633
 - `\start@align`
. 44, 44, 1422,
1462, 1467,
1472, 1477,
1482, 1487,
1502, 1507, 1512
 - `\start@aligned` . 38,
1158, 1215, 1222
 - `\start@gather`
. 1237, 1250, 1257
 - `\start@multiline` ...
. 2236, 2253, 2258
 - `\std@equal` 709, 712, 715
 - `\std@minus` 708, 711, 714
 - `\stepcounter`
... 36, 1564, 2387
 - `stix` package . 4, 4, 4, 4
 - `\strut` 14, 21, 686
 - `\strut@` 15,
511, 1088, 1181,
1232, 1272,
1277, 1298,
1539, 1562,
2047, 2121,
2280, 2299, 2342
 - `\strutbox` 516
 - `\strutbox@`
35, 511, 1086, 1087
 - `subarray` environment
23, 23, 23, 23, 763
 - `\subarray` 775
 - `\substack` 23, 23, 23, 775
 - `\sum` 213, 214
 - `\sum@` 213, 214
- T**
- `\tabskip` 50,
50, 50, 50, 58,
58, 67, 1183,
1187, 1269,
1276, 1279,
1526, 1612,
2051, 2056,
2268, 2576, 2597
 - `\tag` 23,
29, 29, 29, 29,
29, 29, 29, 29,
29, 29, 29, 29,
29, 30, 30, 30,
30, 31, 31, 31,
31, 32, 32, 33,
38, 38, 66, 66,
73, 929, 930,
976, 1263, 1520,
2118, 2264,
2267, 2483, 2502
 - `\tag*` 29, 30
 - `\tag@false`
.. 1079, 1290,
1608, 2265,
2504, 2521, 2611
 - `\tag@help` . 29, 922, 928
 - `\tag@in@align`
..... 71, 973,
1263, 1520, 2264
 - `\tag@in@display` ...
..... 2480, 2502
 - `\tag@in@display@a` .
..... 2480, 2481
 - `\tag@lengths`
.. 1037, 1043,
1112, 1287, 1603

- `\tag@shifts` . 1036, 1055, 1724, 1758, 1823, 1920
`\tag@true` 978, 1533, 1550, 2103, 2382, 2383, 2485, 2611
`\tag@width` . . 1036, 1328, 1730, 1749, 1762, 1788, 1827, 1853, 1924, 1943
`\tagform@` 26, 29, 30, 30, 30, 865, 932, 937, 939, 941, 946, 996, 1004, 1565, 2388, 2488, 2525
`\tagshift@` 52, 1030, 1701, 1702, 1705, 1708, 1711, 2017, 2030
`\tagsleft@false` 29, 53, 55
`\tagsleft@true` 28, 52, 57
`\tagwidth@` 58, 59, 1033, 1328, 1331, 1332, 1348, 1351, 1355, 1356, 1788, 1791, 1803, 1853, 1855, 1870, 1943, 1946, 1962, 1986, 2393, 2395, 2425, 2446
`\tbinom` 7, 182
`\tbtags option` 65
`\tdots@` 350, 437
`\text` 4
`\textellipsis` . 345, 348
`\textfont` 34, 505, 628, 632, 1031
`\textstyle` 23, 193, 249, 461
`\textup` 26, 865
`\tfrac` 6, 6, 179
`\thedots@` 301, 302, 358, 359, 360, 361, 363, 372, 375, 376, 379, 381, 383
`\theequation` . . 26, 849, 852, 937, 939, 996, 1004, 1565, 2388, 2525
`\theparentequation` 2185, 2567, 2571, 2588, 2594
`\thetag` 30, 941
`\thickmuskip` 120, 122, 717
`\thickspace` . . 121, 779
`\thinmuskip` 113, 115, 284, 717
`\thinspace` 114
`\thr@@` 480, 768, 1625, 1639, 1641, 1642, 2139, 2143, 2144, 2153, 2157, 2169, 2187, 2191, 2192, 2196, 2197
`\Tilde` 20, 667
`\tilde` 20, 558, 667
`\tmspace` 5, 111, 113, 115, 117, 119, 120, 122
`\too@wide` 2535, 2538, 2551
`\totwidth@` 50, 50, 52, 58, 1034, 1110, 1252, 1286, 1295, 1296, 1307, 1310, 1311, 1386, 1387, 1489, 1632, 1633, 1646, 1691, 1701, 1714, 1715, 1716, 1743, 1944, 2111, 2269, 2271, 2274, 2370, 2374, 2394
`\try@load@fontshape` 81
`\tw@` . 330, 414, 479, 521, 536, 539, 647, 648, 695, 696, 749, 754, 766, 767, 1031, 1167, 1349, 1360, 1423, 1427, 1430, 1482, 1507, 1512, 1624, 1655, 1661, 1676, 1693, 1704, 1782, 1807, 1847, 1859, 1874, 1883, 1950, 1965, 1977, 1983, 2006, 2137, 2147, 2185, 2567, 2571, 2588, 2594
`\typeout` 56, 66
- U**
- `\uccode` 291, 296, 303, 308, 313, 319, 324, 332
`\underarrow@` 736, 740, 742, 744
`\underleftarrow` . . 741
`\underlefterightarrow` 743
`\underrightarrow` . 739
`\underset` 21, 690
`\unhbox` 280, 754, 1639, 2147, 2552, 2562, 2580, 2601
`\unhcopy` 2135, 2183, 2568, 2591
`\unpenalty` 1630
`\unskip` 350, 934, 1639, 2132, 2136, 2138, 2147, 2184, 2186, 2552, 2562, 2568, 2580, 2591, 2601
`\unvbox` 1630
`\unvcopy` 2130, 2176, 2179
`\uproot` 226, 230, 233, 243, 245
`\uproot@` 227, 233, 245, 246, 254
`\use@mathgroup` . . 589
`\usefont` 105
- V**
- `\value` 850, 855
`\varDelta` 260
`\varGamma` 259
`\varLambda` 262
`\varOmega` 269
`\varPhi` 267
`\varPi` 264
`\varPsi` 268
`\varSigma` 265
`\varTheta` 261
`\varUpsilon` 266
`\varXi` 263
`\vbox` 23, 23, 23, 23
`\vcenter` 23, 23, 525, 764, 776, 1164, 1228, 1436, 2108, 2144, 2163, 2192, 2202
`\Vec` 674

<code>\vec</code> 565, 674	2370, 2374,	<code>xalignat*</code> environ-
<code>\veqno</code> . 2473, 2474, 2527	2393, 2453, 2463	ment 1461
<code>\Vert</code> 5, 142	<code>\widehat</code> 16, 572	<code>\xatlevel@</code> . 44, 44,
<code>\vert</code> 5, 127	<code>\widetilde</code> 16, 571	48, 1417 , 1423,
<code>\vtop</code> 63,	<code>\write</code> 31	1427, 1624,
737, 1164, 1228,		1626, 1647,
1392, 2032,		1704, 1807,
2129, 2176, 2336		1859, 1874,
	X	1950, 1965, 1983
W	<code>\x@calc@shift@lc</code> ..	<code>\xi</code> 19
<code>\wd</code> 255, 648,	... 60 , 1925 , 1933	<code>\xleaders</code> 828
651, 653, 698,	<code>\x@calc@shift@lf</code> ..	<code>\xleftarrow</code> . . . 21, 762
754, 1632, 1642,	... 53 , 53 ,	<code>\xrightarrow</code> 21, 745, 761
2143, 2144,	53 , 54 , 1731, 1739	<code>xxalignat</code> environ-
2153, 2191,	<code>\x@calc@shift@rc</code> ..	ment . . . 44,
2192, 2196,	... 1828 , 1836	44 , 48 , 49 , 51 , 1461
2197, 2308,	<code>\x@calc@shift@rf</code> ..	<code>\xxat@true</code> 1428
2452, 2462, 2586	... 54 , 1763 , 1771	
<code>\wdz@</code> ... 1043, 1295,	<code>\x@lcalc@width</code>	Z
1296, 1339,	... 60 , 1940 , 1996	<code>\z@</code> 7, 7, 13, 28
1347, 1350,	<code>\x@rcalc@width</code> . 54,	
1364, 1586,	1778, 1843, 1901	
	<code>xalignat</code> environment	
	.. 44 , 44 , 51 , 1461	