

The `amsmath` package

Frank Mittelbach Rainer Schöpf Michael Downes
David M. Jones

Version 2.13, 2000/07/18

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 \LaTeX_{2e} , 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}[2000/07/18 v2.13 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\[\do\]\do^\do\_relax
18 \catcode'\^=7 \catcode'\_ =8
```

3 Declare some options

Handling of limits on integrals, sums, operatornames.

```
19 \DeclareOption{intllimits}{\let\ilimits@\displaylimits}
20 \DeclareOption{nointlimits}{\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 L^AT_EX distribution now, i.e., all L^AT_EX 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, \TeX 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 `amsl doc.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

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

```

90 \def\@amsmath@err{\PackageError{amsmath}}

```

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

```

91 \providecommand{\AmS}{\protect\AmSfont
92 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 \LaTeX installation the combination `cmsy/bx/n` does not have a font definition, and the user would get a font substitution warning on screen.

```

93 \newcommand{\AmSfont}{%
94 \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`).

```

95 \def\@mathmeasure#1#2#3{\setbox#1\hbox{\frozen@everymath\@emptytoks
96 \m@th#2#3}}

```

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

```
97 \ifundefined{inf@bad}{%
98   \newcount\inf@bad \inf@bad=1000000 \relax
99 }{}
```

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`.

```
100 \DeclareRobustCommand{\tmspace}[3]{%
101   \ifmmode\mskip#1#2\else\kern#1#3\fi\relax}
102 \renewcommand{\,}{\tmspace+\thinmuskip{.1667em}}
103 \let\thinspace\,
104 \renewcommand{\!}{\tmspace-\thinmuskip{.1667em}}
105 \let\negthinspace\!
106 \renewcommand{\:}{\tmspace+\medmuskip{.2222em}}
107 \let\medspace\:
108 \newcommand{\negmedspace}{\tmspace-\medmuskip{.2222em}}
109 \renewcommand{\;}{\tmspace+\thickmuskip{.2777em}}
110 \let\thickspace\;
111 \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.

```
112 \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.

```
113 \def\@tempa#1#2\@nil{%
114   \ifx\delimiter#1\@tempcnta#2\relax\else\@tempcnta\z@\fi
115 }
116 \exp\@tempa\vert\@empty\@nil
117 \ifnum\@tempcnta>\z@
118   \advance\@tempcnta "4000000
119   \xdef\lvert{\delimiter\number\@tempcnta\space }
120   \advance\@tempcnta "1000000
121   \xdef\rvert{\delimiter\number\@tempcnta\space }
122 \else
123   \ifx\@undefined\lvert
124     % Fall back to cmex encoding since we don't know what else to do.
125     \DeclareMathDelimiter{\lvert}
126       {\mathopen}{symbols}{"6A}{largesymbols}{"0C}
127     \DeclareMathDelimiter{\rvert}
128       {\mathclose}{symbols}{"6A}{largesymbols}{"0C}
129   \fi
130 \fi
131 \exp\@tempa\Vert\@empty\@nil
132 \ifnum\@tempcnta>\z@
133   \advance\@tempcnta "4000000
134   \xdef\lVert{\delimiter\number\@tempcnta\space }
135   \advance\@tempcnta "1000000
```

```

136 \xdef\rVert{\delimiter\number\@tempcnta\space }
137 \else
138 \ifx\@undefined\lVert
139 \DeclareMathDelimiter{\lVert}
140 {\mathopen}{symbols}{"6B}{largesymbols}{"OD}
141 \DeclareMathDelimiter{\rVert}
142 {\mathclose}{symbols}{"6B}{largesymbols}{"OD}
143 \fi
144 \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.

```

145 \@saveprimitive\over\@over
146 \@saveprimitive\atop\@atop
147 \@saveprimitive\above\@above
148 \@saveprimitive\overwithdelims\@overwithdelims
149 \@saveprimitive\atopwithdelims\@atopwithdelims
150 \@saveprimitive\abovewithdelims\@abovewithdelims

```

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

```

151 \DeclareRobustCommand{\primfrac}[1]{%
152 \PackageWarning{amsmath}{%
153 Foreign command \@backslashchar#1;\MessageBreak
154 \protect\frac\space or \protect\genfrac\space should be used instead%
155 \MessageBreak
156 }
157 \global\@xp\let\csname#1\@xp\endcsname\csname @#1\endcsname
158 \csname#1\endcsname
159 }
160 \renewcommand{\over}{\primfrac{over}}
161 \renewcommand{\atop}{\primfrac{atop}}
162 \renewcommand{\above}{\primfrac{above}}
163 \renewcommand{\overwithdelims}{\primfrac{overwithdelims}}
164 \renewcommand{\atopwithdelims}{\primfrac{atopwithdelims}}
165 \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).

```

166 \DeclareRobustCommand{\frac}[2]{\begingroup\@over#2}
167 \newcommand{\dfrac}{\genfrac{}{}{0}}
168 \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.

```

169 \DeclareRobustCommand{\binom}{\genfrac(){}{\z@}}
170 \newcommand{\dbinom}{\genfrac(){}{0}}
171 \newcommand{\tbinom}{\genfrac(){}{1}}

```

`\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).

```

172 \DeclareRobustCommand{\genfrac}[4]{%
173   \def\@tempa{#1#2}%
174   \edef\@tempb{\@nx\@genfrac\@mathstyle{#4}%
175     \csname @@\ifx @#3@over\else above\fi
176     \ifx\@tempa@empty \else withdelims\fi\endcsname}
177   \@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.

```

178 \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`.

```

179 \def\@mathstyle#1{%
180   \ifx\@empty#1\@empty\relax
181   \else\ifcase#1\displaystyle % case 0
182     \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.

```

183 \begingroup
184 \edef\@tempa{\string\mathchar"}
185 \def\@tempb#1"#2\@nil{#1"}
186 \edef\@tempc{\expandafter\@tempb\meaning\coprod "\@nil}
187 \ifx\@tempa\@tempc
188   \global\let\coprod@\coprod
189   \gdef\coprod{\DOTSB\coprod@\slimits@}
190   \global\let\bigvee@\bigvee
191   \gdef\bigvee{\DOTSB\bigvee@\slimits@}
192   \global\let\bigwedge@\bigwedge
193   \gdef\bigwedge{\DOTSB\bigwedge@\slimits@}
194   \global\let\biguplus@\biguplus
195   \gdef\biguplus{\DOTSB\biguplus@\slimits@}
196   \global\let\bigcap@\bigcap
197   \gdef\bigcap{\DOTSB\bigcap@\slimits@}
198   \global\let\bigcup@\bigcup
199   \gdef\bigcup{\DOTSB\bigcup@\slimits@}
200   \global\let\prod@\prod
201   \gdef\prod{\DOTSB\prod@\slimits@}
202   \global\let\sum@\sum
203   \gdef\sum{\DOTSB\sum@\slimits@}
204   \global\let\bigotimes@\bigotimes
205   \gdef\bigotimes{\DOTSB\bigotimes@\slimits@}
206   \global\let\bigoplus@\bigoplus
207   \gdef\bigoplus{\DOTSB\bigoplus@\slimits@}
208   \global\let\bigodot@\bigodot
209   \gdef\bigodot{\DOTSB\bigodot@\slimits@}

```

```

210 \global\let\bigscup@\bigscup
211 \gdef\bigscup{\DOTSB\bigscup@\slimits@}
212 \fi
213 \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]

```

214 \newcommand{\leftroot}{\@amsmath@err{\Invalid@\leftroot}\@eha}
215 \newcommand{\uproot}{\@amsmath@err{\Invalid@\uproot}\@eha}
216 \newcount\uproot@
217 \newcount\leftroot@
218 \renewcommand{\root}{\relaxnext@
219 \DN@{\ifx\@let@token\uproot\let\next@nextii@else
220 \ifx\@let@token\leftroot\let\next@nextiii@else
221 \let\next@plainroot@\fi\fi\next@}%
222 \def\nextii@\uproot##1{\uproot@##1\relax\FN@nextiv@}%
223 \def\nextiv@{\ifx\@let@token\@sptoken\DN@.\{FN@nextv@\}else
224 \DN@.\{FN@nextv@\}\fi\next@.}%
225 \def\nextv@{\ifx\@let@token\leftroot\let\next@nextvi@else
226 \let\next@plainroot@\fi\next@}%
227 \def\nextvi@\leftroot##1{\leftroot@##1\relax\plainroot@}%
228 \def\nextiii@\leftroot##1{\leftroot@##1\relax\FN@nextvii@}%
229 \def\nextvii@{\ifx\@let@token\@sptoken
230 \DN@.\{FN@nextviii@\}else
231 \DN@.\{FN@nextviii@\}\fi\next@.}%
232 \def\nextviii@{\ifx\@let@token\uproot\let\next@nextix@else
233 \let\next@plainroot@\fi\next@}%
234 \def\nextix@\uproot##1{\uproot@##1\relax\plainroot@}%
235 \bgroup\uproot@\z@\leftroot@\z@\FN@next@}
236 \def\plainroot@#1\of#2{\setbox\rootbox\hbox{%
237 $\m@th\scriptscriptstyle{#1}$}%
238 \mathchoice{\r@@t\displaystyle{#2}}{\r@@t\textstyle{#2}}
239 {\r@@t\scriptstyle{#2}}{\r@@t\scriptscriptstyle{#2}}\egroup}

```

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

```

240 \@ifundefined{sqrtsign}{\let\sqrtsign\@@sqrt}{}
241 \def\r@@t#1#2{\setboxz@h{\m@th#1\sqrtsign{#2}$}%
242 \dimen@ht\z@\advance\dimen@-\dp\z@
243 \setbox\@ne\hbox{\m@th#1\mskip\uproot@ mu$}%
244 \advance\dimen@ by1.667\wd\@ne
245 \mkern-\leftroot@ mu\mkern5mu\raise.6\dimen@\copy\rootbox
246 \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.

```

247 \@ifundefined{varGamma}{%
248 \DeclareMathSymbol{\varGamma}{\mathord}{letters}{"00}
249 \DeclareMathSymbol{\varDelta}{\mathord}{letters}{"01}
250 \DeclareMathSymbol{\varTheta}{\mathord}{letters}{"02}
251 \DeclareMathSymbol{\varLambda}{\mathord}{letters}{"03}
252 \DeclareMathSymbol{\varXi}{\mathord}{letters}{"04}
253 \DeclareMathSymbol{\varPi}{\mathord}{letters}{"05}
254 \DeclareMathSymbol{\varSigma}{\mathord}{letters}{"06}
255 \DeclareMathSymbol{\varUpsilon}{\mathord}{letters}{"07}
256 \DeclareMathSymbol{\varPhi}{\mathord}{letters}{"08}

```



```

257 \DeclareMathSymbol{\varPsi}{\mathord}{letters}{"09}
258 \DeclareMathSymbol{\varOmega}{\mathord}{letters}{"0A}
259 }{}

```

$\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\mathcal{T}\mathcal{E}\mathcal{X}$ redefines `\overline` as shown here, for reasons that are probably less important in $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$: Make it read its argument as a macro argument rather than a “math field” (*The $\mathcal{T}\mathcal{E}\mathcal{X}$ book*, 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.

```

260 \@saveprimitive\overline\@@overline
261 \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.

```

262 \newcommand{\boxed}[1]{\fbox{\m@th$\displaystyle#1$}}
263 \newcommand{\implies}{\DOTSB\;\Longrightarrow\;}
264 \newcommand{\impliedby}{\DOTSB\;\Longleftarrow\;}
265 \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`).

```

266 \newcommand{\nobreakdash}{\leavevmode
267 \toks@{\@emptytoks \def\@tempa##1{\toks@{\xp{\the\toks@-}\FN@next@}}%
268 \DN@{\ifx\@let@token-\@xp\@tempa
269 \else\setboxz@h{\the\toks@\nobreak}\unhboxz@\fi}%
270 \FN@next@
271 }

```

`\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.

```

272 \renewcommand{\colon}{\nobreak\mskip2mu\mathpunct{}}\nonscript
273 \mkern-\thinmuskip{:}\mskip6mu\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`.

```

274 \let\ifgtest@\iffalse % initial value
275 \def\gtest@true{\global\let\ifgtest@\iftrue}
276 \def\gtest@false{\global\let\ifgtest@\iffalse}
277 \let\DOTSI\relax
278 \let\DOTSB\relax
279 \let\DOTSX\relax
280 {\uccode'7='\ \uccode'8='m \uccode'9='a \uccode'0='t \uccode'!='h
281 \uppercase{%
282 \gdef\math@#1#2#3#4#5#6\math@{\gtest@false\ifx 7#1\ifx 8#2%
283 \ifx 9#3\ifx 0#4\ifx !#5\xdef\meaning@{#6}\gtest@true
284 \fi\fi\fi\fi\fi}}
285 {\uccode'7='c \uccode'8='h \uccode'9='\ "
286 \uppercase{\gdef\mathch@#1#2#3#4#5#6\mathch@{\gtest@false
287 \ifx 7#1\ifx 8#2\ifx 9#5\gtest@true\xdef\meaning@{#6}\fi\fi\fi}}
288 \newcount\classnum@

```

```

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

```
332 \@ifundefined{ldots}{\def\ldots{\mathellipsis}}{}
```

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

```

333 \DeclareRobustCommand{\ldots}{%
334 \ifmmode \mathellipsis \else \textellipsis \fi
335 }
336 \DeclareRobustCommand{\dots}{%
337 \ifmmode \exp\mdots@else \exp\textellipsis \fi
338 }
339 \def\tdots@{\leavevmode\unskip\relaxnext@
340 \DN@{\$m@th@ldots\,
341 \ifx\@let@token,\,$\else\ifx\@let@token.\,$\else
342 \ifx\@let@token;\,$\else\ifx\@let@token:\,$\else

```


The `\doteq` command formerly used `\buildrel`; we avoid that because it requires ‘`\over`’ as part of its syntax. Use `0pt` instead of `\z@` for robustitude.

```
449 \renewcommand{\doteq}{%
450   \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`.

```
451 \newif\if@display
452 \everydisplay@xp{\the\everydisplay \@displaytrue}
```

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

```
453 \renewcommand{\int}{\DOTSI\intop\ilimits@}
454 \renewcommand{\oint}{\DOTSI\ointop\ilimits@}
455 \def\intkern@{\mkern-6mu\mathchoice{\mkern-3mu}{-}{-}}
456 \def\intdots@{\mathchoice{\@cdots}%
457   {{\cdotp}\mkern1.5mu{\cdotp}\mkern1.5mu{\cdotp}}%
458   {{\cdotp}\mkern1mu{\cdotp}\mkern1mu{\cdotp}}%
459   {{\cdotp}\mkern1mu{\cdotp}\mkern1mu{\cdotp}}}
460 %
461 \newcommand{\iint}{\DOTSI\protect\MultiIntegral{2}}
462 \newcommand{\iiint}{\DOTSI\protect\MultiIntegral{3}}
463 \newcommand{\iiiiint}{\DOTSI\protect\MultiIntegral{4}}
464 \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.

```
465 \newcommand{\MultiIntegral}[1]{%
466   \edef\ints@c{\noexpand\intop
467     \ifnum#1=\z@\noexpand\intdots@ \else\noexpand\intkern@\fi
468     \ifnum#1>\tw@\noexpand\intop\noexpand\intkern@\fi
469     \ifnum#1>\thr@\noexpand\intop\noexpand\intkern@\fi
470     \noexpand\intop
471     \noexpand\ilimits@
472   }%
473   \futurelet\@let@token\ints@a
474 }

475 \def\ints@a{%
476   \ifx\limits\@let@token \ints@b
477   \else \ifx\displaylimits\@let@token \ints@b
478   \else \ifx\ilimits@\displaylimits \ints@b
479   \fi\fi\fi
480   \ints@c
481 }

482 \def\ints@b{%
483   \mkern-7mu\mathchoice{\mkern-2mu}{-}{-}%
484   \mathop\bgroup
485     \mkern7mu\mathchoice{\mkern2mu}{-}{-}%
486     \let\ilimits@\egroup
487 }%
```

9 Size dependent definitions

We now define all stuff which has to change whenever a new math size is to be activated. L^AT_EX 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@ 488 \newbox\Mathstrutbox@
\resetMathstrut@ 489 \setbox\Mathstrutbox@=\hbox{}
490 \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 `$` after a font size change, we want to avoid using another math formula `$. . .$` 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.

```
491 \def\resetMathstrut@{%
492   \setbox\z@\hbox{%
493     \mathchardef\@tempa\mathcode'\(\relax
494     \def\@tempb##1"##2##3{\the\textfont"##3\char"}%
495     \expandafter\@tempb\meaning\@tempa \relax
496   }%
```

These height and depth assignments are implicitly global.

```
497   \ht\Mathstrutbox@\ht\z@ \dp\Mathstrutbox@\dp\z@
498 }
499 \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]

```
500 \newbox\strutbox@
501 \def\strut@{\copy\strutbox@}
502 \def\reset@strutbox@{%
```

```

503 \global\setbox\strutbox@\hbox{%
504   \lower.5\normallineskiplimit
505   \vbox{\kern-\normallineskiplimit\copy\strutbox}}
506 \addto@hook\every@math@size{\reset@strutbox@}
507 \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.

```

508 \renewcommand{\big}{\bBigg@\@ne}
509 \renewcommand{\Big}{\bBigg@{1.5}}
510 \renewcommand{\bigg}{\bBigg@\tw@}
511 \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.

```
512 \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.

```

513   {\@mathmeasure\z@{\nulldelimiterspace\z@}%
514    {\left#2\center to#1\big@size{\right.}%
515     \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@`.

```

516 \addto@hook\every@math@size{%
517   \global\big@size 1.2\ht\Mathstrutbox@
518   \global\advance\big@size 1.2\dp\Mathstrutbox@ }
519 \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.

```

520 \def\accentclass@{7}
521 \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.

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

The next one corrects the `\cal` alphabet.

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

Triple and quadruple dot accents.

```

524 \newcommand{\dddot}[1]{%
525   {\mathop{#1}\limits^{\vbox to-1.4\ex@{\kern-\tw@ex@
526     \hbox{\normalfont ...}\vss}}}}

```

```

527 \newcommand{\ddddot}[1]{%
528   {\mathop{#1}\limits^{\vbox to-1.4\ex@{\kern-\tw@ex@
529     \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`.

```

530 \def\set@mathaccent#1#2#3#4{%
531   \xdef#2{\@nx\protect\@nx\mathaccentV
532     {\@xp@gobble\string#2}\hexnumber@#1#4}%
533 }

```

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.

```

534 \def\@tempa#1{\@xp\@tempb\meaning#1\@nil#1}
535 \def\@tempb#1>#2#3 #4\@nil#5{%
536   \@xp\ifx\csname#3\endcsname\mathaccent
537     \@tempc#4?"7777\@nil#5%
538   \else
539     \PackageWarningNoLine{amsmath}{%
540       Unable to redefine math accent \string#5}%
541   \fi
542 }
543 \def\@tempc#1"#2#3#4#5#6\@nil#7{%
544   \chardef\@tempd"#3\relax\set@mathaccent\@tempd{#7}{#2}{#4#5}}
545 \@tempa{\hat}
546 \@tempa{\check}
547 \@tempa{\tilde}
548 \@tempa{\acute}
549 \@tempa{\grave}
550 \@tempa{\dot}
551 \@tempa{\ddot}
552 \@tempa{\breve}
553 \@tempa{\bar}
554 \@tempa{\vec}
555 \@ifundefined{mathring}{%
556   \DeclareMathAccent{\mathring}{\mathalpha}{operators}{"17}
557 }{%
558   \@tempa{\mathring}
559 }
560 %%\@tempa\widetilde
561 %%\@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 L^AT_EX 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.

```
562 \newcommand{\acc@check}{}
563 \newcommand{\acc@error}{}
564 \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.

```
565 \def\acc@error{%
566   \@amsmath@err{%
567     Improper argument for math accent:\MessageBreak
568     Extra braces must be added to prevent wrong output%
569   }\@ehc
570 }
```

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.

```
571 \newcount\macc@depth
```

Provide this function in case it is not already available.

```
572 \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.

```
573 \def\mathaccentV#1#2#3#4#5{%
574   \ifmmode
575     \gdef\macc@tmp{\macc@depth\@ne}%
576     \setbox\z@\hbox{%
577       \let\mathaccentV\macc@test
578       \let\use@mathgroup\@gobbletwo \let\select@group\@gobblethree
579       \frozen@everymath{}\$#5$%
580     }%
581     \macc@tmp
582     \ifnum\macc@depth=\@ne
583       \global\let\macc@nucleus\@empty
584       \mathaccent"\accentclass@
585     \else
586       \@xp\macc@nested
587     \fi
588     #2#3#4{#5}%
589     \macc@nucleus
590   \else
591     \@xp\nonmatherr@\csname#1\endcsname
592   \fi
593 }
594 \def\macc@test#1#2#3#4{\xdef\macc@tmp{\macc@tmp\advance\macc@depth\@ne}}
595 \def\macc@group{-1}
596 \def\macc@nested#1#2#3#4{%
```

```

597 \begingroup
598 \let\math@bgroup\@empty \let\math@egroup\macc@set@skewchar
599 \mathsurround\z@ \frozen@everymath{\mathgroup\macc@group\relax}%
600 \macc@set@skewchar\relax
601 \let\mathaccentV\macc@nested@a
602 \macc@nested@a\relax#1#2#3{#4}%
603 \endgroup
604 }

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

```

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

```

607 \ifnum\macc@group=\mathgroup
608 \else \macc@set@skewchar\relax \edef\macc@group{\the\mathgroup}%
609 \fi
610 \mathchardef\macc@code "\accentclass@ #2#3#4\relax
611 \macc@palette\macc@a{#5}%
612 }

```

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.

```

613 \def\macc@set@skewchar#1{%
614 \begingroup
615 \ifnum\mathgroup=\m@ne \let\@tempa\@ne
616 \else
617 \ifnum\skewchar\textfont\mathgroup=\m@ne \let\@tempa\@ne
618 \else \let\@tempa\mathgroup
619 \fi
620 \fi
621 \count@=\skewchar\textfont\@tempa
622 \advance\count@"7100
623 \edef\@tempa{\endgroup
624 \mathchardef\noexpand\macc@skewchar=\number\count@\relax}%
625 \@tempa
626 #1%
627 }

```

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.

```

628 \def\macc@a#1#2{%
629 \begingroup
630 \let\macc@style#1\relax
631 \def\macc@palette##1{##1\macc@style}%
632 \advance\macc@depth\m@ne
633 \ifnum\macc@depth=\z@
634 \gdef\macc@nucleus{#2}%

```

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

```

635 \setbox\z@\hbox{##1#2\@empty}\macc@skewchar$}%
636 \setbox\tw@\hbox{##1#2\@empty\macc@skewchar$}%
637 \dimen@tw@\wd\tw@ \advance\dimen@-\tw@\wd\z@
638 \xdef\macc@kerna{\the\dimen@\relax}%
639 \setbox4\hbox{##1#2\acc@check\@empty$}%
640 \global\setbox\@ne\hbox to\wd4{}%
641 \ht\@ne\ht4 \dp\@ne\dp4
642 \xdef\macc@kernb{\the\wd4\relax}%
643 \mathaccent\macc@code{\box\@ne\kern\macc@kerna}%
644 \else
645 \mathaccent\macc@code{\let\macc@adjust\@empty #1#2\@empty}%
646 \macc@adjust
647 \fi
648 \endgroup
649 }

650 \def\macc@adjust{%
651 \dimen@\macc@kerna\advance\dimen@\macc@kernb
652 \kern-\dimen@
653 }

```

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

```

654 \def\Hat{\hat}
655 \def\Check{\check}
656 \def\Tilde{\tilde}
657 \def\Acute{\acute}
658 \def\Grave{\grave}
659 \def\Dot{\dot}
660 \def\Ddot{\ddot}
661 \def\Breve{\breve}
662 \def\Bar{\bar}
663 \def\Vec{\vec}

```

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

```

664 \def\nonmatherr@#1{\@amsmath@err{\protect
665 #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]

```

666 \renewcommand{\bmod}{\nonscript\mskip-\medmuskip\mkern5mu\mathbin
667 {\operator@font mod}\penalty900
668 \mkern5mu\nonscript\mskip-\medmuskip}
669 \newcommand{\pod}[1]{\allowbreak
670 \if@display\mkern18mu\else\mkern8mu\fi(#1)}
671 \renewcommand{\pmod}[1]{\pod{\operator@font mod}\mkern6mu#1}}
672 \newcommand{\mod}[1]{\allowbreak\if@display\mkern18mu
673 \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.

```

674 \newcommand{\cfrac}[3][c]{\displaystyle\frac{
675 \strut\ifx r#1\hfill\fi#2\ifx l#1\hfill\fi}{#3}}%
676 \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.

```
677 \newcommand{\overset}[2]{\binrel@{#2}%
678   \binrel@{\mathop{\kern\z@#2}\limits^{#1}}}}
679 \newcommand{\underset}[2]{\binrel@{#2}%
680   \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.

```
681 \newcommand{\sideset}[3]{%
682   \@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.

```
683   \global\setbox\@ne\vbox to\ht\z@{\dp\@ne\dp\z@
684     \setbox\tw@\box\@ne
685     \@mathmeasure4\displaystyle{\copy\tw@#1}%
686     \@mathmeasure6\displaystyle{#3\nolimits#2}%
687     \dimen@-\wd6 \advance\dimen@\wd4 \advance\dimen@\wd\z@
688     \hbox to\dimen@{\mathop{\kern-\dimen@\box4\box6}}%
689 }
```

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

```
690 \renewcommand{\smash}[1][tb]{%
691   \def\mb@t{\ht}\def\mb@b{\dp}\def\mb@tb{\ht\z@\z@\dp}%
692   \edef\finsh{\csname mb@#1\endcsname\z@\z@ \box\z@}%
693   \ifmode \@xp\mathpalette\@xp\mathsm@sh
694     \else \@xp\makesm@sh
695     \fi
696 }
```

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.

```
697 \mathchardef\std@minus\mathcode'\-\relax
698 \mathchardef\std@equal\mathcode'\=\relax
```

In case some alternative math fonts are loaded later:

```
699 \AtBeginDocument{%
700   \mathchardef\std@minus\mathcode'\-\relax
701   \mathchardef\std@equal\mathcode'\=\relax
702 }
703 \def\relbar{\mathrel{\mathpalette\mathsm@sh\std@minus}}
704 \def\Relbar{\mathrel\std@equal}
```

```

705 \def\arrowfill@#1#2#3#4{%
706   $\m@th\thickmuskip0mu\medmuskip\thickmuskip\thinmuskip\thickmuskip
707   \relax#4#1\mkern-7mu%
708   \cleaders\hbox{$#4\mkern-2mu#2\mkern-2mu$}\hfill
709   \mkern-7mu#3$%
710 }
711 \def\leftarrowfill@{\arrowfill@\leftarrow\relbar\relbar}
712 \def\rightarrowfill@{\arrowfill@\rightarrow\relbar\relbar}
713 \def\leftrightharrowfill@{\arrowfill@\leftarrow\relbar\rightarrow}
714 \def\Leftarrowfill@{\arrowfill@\Leftarrow\Relbar\Relbar}
715 \def\Rightarrowfill@{\arrowfill@\Rightarrow\Relbar\Relbar}
716 \def\Leftrightarrowfill@{\arrowfill@\Leftrightarrow\Relbar\Relbar}

717 \def\overarrow@#1#2#3{\vbox{\ialign{##\crcr#1#2\crcr
718   \noalign{\nointerlineskip}$\m@th\hfil#2#3\hfil$\crcr}}
719 \renewcommand{\overrightarrow}{%
720   \mathpalette{\overarrow@\rightarrowfill@}}
721 \renewcommand{\overleftarrow}{%
722   \mathpalette{\overarrow@\leftarrowfill@}}
723 \newcommand{\overleftrightharrow}{%
724   \mathpalette{\overarrow@\leftrightharrowfill@}}

725 \def\underarrow@#1#2#3{%
726   \vtop{\ialign{##\crcr$\m@th\hfil#2#3\hfil$\crcr
727   \noalign{\nointerlineskip\kern1.3\ex@}#1#2\crcr}}
728 \newcommand{\underrightarrow}{%
729   \mathpalette{\underarrow@\rightarrowfill@}}
730 \newcommand{\underleftarrow}{%
731   \mathpalette{\underarrow@\leftarrowfill@}}
732 \newcommand{\underleftrightharrow}{%
733   \mathpalette{\underarrow@\leftrightharrowfill@}}

734 %\newcommand{\xrightarrow}[2][\ext@arrow 0359\rightarrowfill@{#1}{#2}]
735 \def\ext@arrow#1#2#3#4#5#6#7{%
736   \mathrel{\mathop{#1}

```

Measure the superscript and subscript.

```

737   \setbox\z@\hbox{#5\displaystyle}%
738   \setbox\tw@\vbox{\m@th
739     \hbox{$\scriptstyle\mkern#3mu{#6}\mkern#4mu$}%
740     \hbox{$\scriptstyle\mkern#3mu{#7}\mkern#4mu$}%
741     \copy\z@
742   }%
743   \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.

```

744 \limits
745 \@ifnotempty{#7}{^{\if0#1\else\mkern#1mu\fi
746   #7\if0#2\else\mkern#2mu\fi}}%
747 \@ifnotempty{#6}{_{\if0#1\else\mkern#1mu\fi
748   #6\if0#2\else\mkern#2mu\fi}}%
749 }

```

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

```

750 \newcommand{\xrightarrow}[2][\ext@arrow 0359\rightarrowfill@{#1}{#2}]
751 \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.

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

Note: The predecessors of `subarray` (`Sb` and `Sp`, inherited from $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$) 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`.

```
753 \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!

```
754 \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.

```
755 \baselineskip\fontdimen10 \scriptfont\tw@
756 \advance\baselineskip\fontdimen12 \scriptfont\tw@
757 \lineskip\thr@\fontdimen8 \scriptfont\thr@@
758 \lineskiplimit\lineskip
```

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

```
759 \ialign\bgroup\ifx c#1\hfil\fi
760   $\m@th\scriptstyle##$\hfil\cr
761 }{\%
762 \cr\egroup\egroup
763 }
```

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

```
764 \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 $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$ 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)

```

765 \newenvironment{smallmatrix}{\null\,\vcenter\bgroup
766 \Let@\restore@math@cr@default@tag
767 \baselineskip6\ex@ \lineskip1.5\ex@ \lineskiplimit\lineskip
768 \ialign\bgroup\hfil$\m@th\scriptstyle##$\hfil&&\thickspace\hfil
769 $\m@th\scriptstyle##$\hfil\crcr
770 }{%
771 \crcr\egroup\egroup\,%
772 }

```

`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 L^AT_EX doesn't kill the definitions found in `plain.tex`, even though it probably should because of their foreign syntax.

```

773 \renewenvironment{matrix}{%
774 \matrix@check\matrix\env@matrix
775 }{%
776 \endarray \hskip -\arraycolsep
777 }

```

`\env@matrix`

```

778 \def\env@matrix{\hskip -\arraycolsep
779 \let@ifnextchar\new@ifnextchar
780 \array{* \c@MaxMatrixCols c}}

```

`\c@MaxMatrixCols`

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

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

```

782 \def\matrix@check#1{%
783 \@xp@ifx\c@name\@currenenv\endc@name#1%
784 \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.)

```

785 \@xp@gobble
786 \fi
787 }

```

`\matrix@error`

```
788 \def\matrix@error#1{%
```



```

789 \amsmath@err{%
790 Old form ‘\string#1’ should be \string\begin{\@xp@gobble\string#1}%
791 }{%
792 ‘\string#1{...}’ is old Plain-TeX syntax whose use is
793 ill-advised in LaTeX.%
794 }%
795 }

796 \renewenvironment{pmatrix}{%
797 \left(
798 \matrix@check\pmatrix\env@matrix
799 }{
800 \endmatrix\right)%
801 }
802 \newenvironment{bmatrix}{\left[\env@matrix\endmatrix\right]}
803 \newenvironment{Bmatrix}{%
804 \left\lbracket\env@matrix
805 }{%
806 \endmatrix\right\rbrace
807 }
808 \newenvironment{vmatrix}{\left\lvert\env@matrix\endmatrix\right\rvert}
809 \newenvironment{Vmatrix}{\left\lVert\env@matrix\endmatrix\right\lVert}
810 \let\hdots\@ldots
811 \newcommand{\hdotsfor}[1]{%
812 \ifx[#1\@xp\shdots@for\else\hdots@for\@ne{#1}\fi}
813 \newmuskip\dotsspace@
814 \def\shdots@for#1{\hdots@for{#1}}
815 \def\hdots@for#1#2{\multicolumn{#2}c%
816 {\m@th\dotsspace@1.5mu\mkern-#1\dotsspace@
817 \xleaders\hbox{\$m@th\mkern#1\dotsspace@.\mkern#1\dotsspace@$}%
818 \hfill
819 \mkern-#1\dotsspace@}}%
820 }

```

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`.

```

821 \renewenvironment{cases}{%
822 \matrix@check\cases\env@cases
823 }{%
824 \endarray\right.%
825 }
826 \def\env@cases{%
827 \let\@ifnextchar\new@ifnextchar
828 \left\lbracket
829 \def\arraystretch{1.2}%
830 \array{@{}l@{\quad}l@{}}%
831 }

```

14 Equation sub-numbering

```

832 \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.

```

833 \@ifundefined{ignorespacesafterend}{%
834 \def\ignorespacesafterend{\global\@ignoretrue}%
835 }{ }

```

```
836 \newenvironment{subequations}{%
```

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

```
837 \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`.

```
838 \protected@edef\theparentequation{\theequation}%
```

```
839 \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.

```
840 \setcounter{equation}{0}%
```

```
841 \def\theequation{\theparentequation\alph{equation}}%
```

```
842 \ignorespaces
```

```
843 }{%
```

```
844 \setcounter{equation}{\value{parentequation}}%
```

```
845 \ignorespacesafterend
```

```
846 }
```

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.

```
847 \newcommand{\numberwithin}[3][\arabic]{%
```

```
848 \@ifundefined{c#2}{\@nocounterr{#2}}{%
```

```
849 \@ifundefined{c#3}{\@nocnterr{#3}}{%
```

```
850 \@addtoreset{#2}{#3}}%
```

```
851 \@xp\xdef\csname the#2\endcsname{%
```

```
852 \@xp\@nx\csname the#3\endcsname .\@nx#1{#2}}}%
```

```
853 }
```

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.

```
854 \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.

```
855 \newcount\dspbrk@lv1
```

```
856 \dspbrk@lv1=-1
```

`\interdisplaylinepenalty` We set the `\interdisplaylinepenalty` to 10000.

```
857 \interdisplaylinepenalty\@M
```

`\allowdisplaybreaks` The `\allowdisplaybreaks` command. Since this is intended for use outside displayed formulas (typically in the preamble), it does not need to use `\new@ifnextchar`.

```
858 \newcommand{\allowdisplaybreaks}[1][4]{%
859   \interdisplaylinepenalty\getdsp@pen{#1}\relax
860 }
```

`\getdsp@pen` Modelled after L^AT_EX's `\@getpen`. We use higher numbers than would normally be provided by `\@lowpenalty`, `\@medpenalty`, and `\@highpenalty`, since display breaks are almost always less desirable.

```
861 \def\getdsp@pen#1{%
862   \ifcase #1\@M \or 9999 \or 6999 \or 2999 \or \z@\fi
863 }
```

`\displaybreak` For breaks in a certain row of a alignment.

```
\dspbrk@
\dspbrk@context
nogood@displaybreak
864 \newcommand{\displaybreak}{\new@ifnextchar[\dspbrk@{\dspbrk@[4]}}
865 \chardef\dspbrk@context=\sift@0n
866 \def\dspbrk@[#1]{%
867   \ifmeasuring@
868   \else
869     \ifcase\dspbrk@context % case 0 --- OK
870     \global\dspbrk@lvl #1\relax
871     \or % case 1 --- inside a box
872     \nogood@displaybreak
873     \else % other cases --- outside of a display
874     \@amsmath@err{\Invalid@displaybreak}\@eha
875     \fi
876   \fi
877 }
```

This is the value of `\displaybreak` when it occurs inside some structure where it will not work.

```
878 \def\nogood@displaybreak{%
879   \@amsmath@err{\protect
880   \displaybreak\space cannot be applied here}%
881 {One of the enclosing environments creates an
882   unbreakable box\MessageBreak
883 (e.g., split, aligned, gathered, ...).}%
884 }
```

`\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 T_EX'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).

```
885 \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.

```
886 \@ifstar{\global\@eqpen\@M\math@cr@}%
```

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

```
887   {\global\@eqpen
888     \ifnum\dspbrk@lvl <\z@ \interdisplaylinepenalty
889     \else -\@getpen\dspbrk@lvl \fi
890   \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.

```
891 \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.

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

Finally we put the additional space onto the vertical list.

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

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

```
894 \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.

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

This is also the default case.

```
896 \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.

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

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

```
898 \def\intertext@{%
899 \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.)

```
900 \ifvmode\else\\\@empty\fi
901 \noalign{%
902 \penalty\postdisplaypenalty\vskip\belowdisplayskip
903 \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.

```
904 \ifdim\linewidth=\columnwidth
905 \else \parshape\@ne \@totalleftmargin \linewidth
906 \fi
907 \noindent##1\par}%
908 \penalty\predisplaypenalty\vskip\abovedisplayskip%
909 }%
910 }}
```

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.

```

911 \newhelp\tag@help
912  {tag cannot be used at this point.\space
913   If you don't understand why^^Jyou should consult
914   the documentation.^^JBut don't worry: just continue, and I'll
915   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.

```

916 \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.

```

917 \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.

`\default@tag` 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.

```

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

```

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

```

919 \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.

```

920 \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.)

```

921 \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.)

```

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

```

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

```
923 \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).

```
924 \iftagsleft@
925   \def\@eqnnum{\hbox to1sp{\rlap{\normalfont\normalcolor
926     \hskip -\displaywidth\tagform@\theequation}}}
927 \else
928   \def\@eqnnum{{\normalfont\normalcolor \tagform@\theequation}}
929 \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.

```
930 \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.

```
\make@df@tag
\make@df@tag@@
\make@df@tag@@@
```

```
931 \let\df@tag\@empty
932 \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.

```
933 \def\make@df@tag@#1{%
934   \gdef\df@tag{\maketag@@@{#1}\def\@currentlabel{#1}}}
```

Autogenerated number:

```
935 \def\make@df@tag@@#1{\gdef\df@tag{\tagform@{#1}%
936   \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`.

```
\df@label
```

```
937 \let\ltx@label\label
938 %
939 \def\label@in@display{%
940   \ifx\df@label\@empty\else
941     \@amsmath@err{Multiple \string\label's:
942       label '\df@label' will be lost}\@eha
943   \fi
944   \gdef\df@label
945 }
```

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

```
946 \toks@\exp{\arrayparboxrestore \let\label\ltx@label}%
947 \edef\arrayboxrestore{\the\toks@}
948 \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.

```
949 \def\make@display@tag{%
950   \if@eqnsw \incr@eqnum \print@eqnum
951   \else \iftag@ \df@tag \global\let\df@tag\@empty \fi
952   \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.

```
953   \ifmeasuring@
954   \else
955     \ifx\df@label\@empty
956     \else
957       \exp\ltx@label\exp{\df@label}%
958       \global\let\df@label\@empty
959     \fi
960   \fi
961 }
```

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`.

```
962 \def\tag@in@align{%
963   \relax
964   \iftag@
965     \DN@{\invalid@tag{Multiple \string\tag}}%
966   \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`.

```
967   \global\tag@true
```

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

```
968   \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.

```
969     \let\next@\make@df@tag
```

```

970   \fi
971   \next@
972 }

```

`\raisetag` Usage: `\raisetag` $\langle dimen \rangle$

This will modify the vertical placement of the tag of the current equation by $\langle dimen \rangle$. 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.

```

973 \newcommand{\raisetag}[1]{\skip@#1\relax
974  \xdef\raise@tag{\vskip\iftagsleft@else-\fi\the\skip@\relax}%
975 }

```

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

```

976 \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 T_EX primitive that should not be redefined.

```

977 \newcommand{\notag}{\nonumber}

```

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

```

978 \renewcommand{\nonumber}{%
979   \if@eqnsw
980     \ifx\incr@eqnum\@empty \addtocounter{equation}\m@ne \fi
981   \fi
982   \let\print@eqnum\@empty \let\incr@eqnum\@empty
983   \global\@eqnswfalse
984 }

985 \def\print@eqnum{\tagform@\theequation}
986 \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

`\ifinalign@` We define two switches that are set to true in certain alignments: `inalign@` and `\ifingather@` `ingather@` 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.

```

987 \newif\ifinalign@
988 \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.
    989 \exp\def\exp@arrayparboxrestore\exp{\@arrayparboxrestore
    990 \ingather@false\inalign@false \default@tag
    991 \let\spread@equation\@spread@equation
    992 \let\reset@equation\@empty
    993 \def\print@eqnum{\tagform@theequation}%
    994 \def\incr@eqnum{\refstepcounter{equation}\let\incr@eqnum\@empty}%
    995 }

\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.
    996 \newif\iftag@

\ifst@rred The switch st@rred is set to true by all starred environments and set to false
by the unstarred versions.
    997 \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.
    998 \newif\ifmeasuring@

\ifshifftag@ \ifshifftag@ is used by gather to communicate between \calc@shift@gather
and \place@tag@gather whether an equation tag should be shifted to a separ-
ate line. It’s also used by multiline.
    999 \newif\ifshifftag@

\row@
    1000 \newcount\row@

\column@ The counter \column@ is used by the alignment macros to keep track of the
current column.
    1001 \newcount\column@

\column@plus \column@plus is a useful abbreviation.
    1002 \def\column@plus{%
    1003   \global\advance\column@\@ne
    1004 }

\maxfields@
    1005 \newcount\maxfields@

\add@amp
\add@amps 1006 \def\add@amp#1{\if m#1&\exp\add@amp\fi}
    1007 \def\add@amps#1{%
    1008   \begingroup
    1009   \count@#1\advance\count@-\column@
    1010   \edef\@tempa{\endgroup
    1011     \exp\add@amp\romannumeral\number\count@ 000q}%
    1012   \@tempa
    1013 }

```

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

```

1014 \newhelp\andhelp@
1015 {An extra & here is so disastrous that you should probably exit^^J
1016 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.

```

1017 \newdimen\eqnshift@

```

`\alignsep@`

```

1018 \newdimen\alignsep@

```

`\tagshift@`

```

1019 \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 \TeX 's built-in value.

```

1020 \newcommand{\mintagsep}{.5\fontdimen6\textfont\tw@}

```

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

```

1021 \newcommand{\minalignsep}{10pt}

```

`\tagwidth@`

```

1022 \newdimen\tagwidth@

```

`\totwidth@`

```

1023 \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.

```

1024 \newdimen\lineht@

```

`\tag@width`

```

1025 \def\tag@width#1{%
1026   \ifcase\@xp#1\tag@lengths\fi
1027 }
1028
1029 \def\savetaglength@{%
1030   \begingroup
1031     \let\or\relax
1032     \xdef\tag@lengths{\tag@lengths\or \the\wdz@}%
1033   \endgroup
1034 }
1035
1036 \def\shift@tag#1{%
1037   \ifcase\@xp#1\tag@shifts\fi\relax
1038 }
1039
1040 \let\tag@shifts\@empty

```

`\saveshift@`

```

1041 \def\saveshift@#1{%
1042   \begingroup
1043     \let\or\relax
1044     \xdef\tag@shifts{\or#1\tag@shifts}%
1045   \endgroup
1046 }

```

`\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.

```
1047 \def\spread@equation{\reset@strutbox@
1048   \openup\jot \let\spread@equation\@empty}
1049 \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/Spivak/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.

```
1050 \def\display@{\@display@init{}}
1051 \def\@display@init#1{%
1052   \global\dt@ptrue \spread@equation
1053   \everycr{%
1054     \noalign{%
1055       #1%
1056       \ifdt@p
1057         \global\dt@pfalse
1058         \vskip-\lineskiplimit
1059         \vskip\normallineskiplimit
1060       \else
1061         \penalty\@eqpen \global\dspbrk@lvl@m@ne
1062       \fi
1063     }%
1064   }%
1065 }
```

`\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 `multiline` we don't want that to happen in `\everycr`.

```
1066 \def\display@{\@display@init{%
1067   \global\column@z@ \global\dspbrk@lvl@m@ne
1068   \global\tag@false \global\let\raise@tag\@empty
1069 }}
```

`\black@` This macro is made to produce an overfull 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`.

```
1070 \def\black@#1{%
1071   \noalign{%
1072     \ifdim#1>\displaywidth
1073       \dimen@\prevdepth
1074       \nointerlineskip
1075       \vskip-\ht\strutbox@
1076       \vskip-\dp\strutbox@
1077       \vbox{\noindent\hbox to#1{\strut@\hfill}}}%
1078     \prevdepth\dimen@
1079   \fi
1080   }%
1081 }
```

`\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].

```

1082 \def\savecounters@{%
1083   \begingroup
1084     \def\elt##1{%
1085       \global\csname c@##1\endcsname\the\csname c@##1\endcsname}%
1086     \xdef\gtempa{%
1087       \cl@ckpt
1088       \let\@nx\restorecounters@\@nx\@empty
1089     }%
1090   \endgroup
1091   \let\restorecounters@\gtempa
1092 }
1093 %
1094 \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.

```

1095 \def\savealignstate@{%
1096   \begingroup
1097     \let\or\relax
1098     \xdef\gtempa{%
1099       \global\totwidth\the\totwidth@
1100       \global\row\the\row@
1101       \gdef\@nx\tag@lengths{\tag@lengths}%
1102       \let\@nx\restorealignstate@\@nx\@empty
1103     }%
1104   \endgroup
1105   \let\restorealignstate@\gtempa
1106 }
1107 %
1108 \let\restorealignstate@\@empty

```

`\savecolumn@`
`\restorecolumn@`

```

1109 \def\savecolumn@{%
1110   \edef\restorecolumn@{%
1111     \global\column@\number\column@
1112     \let\@nx\restorecolumn@\@nx\@empty
1113   }%
1114 }
1115 \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.

```
1116 \newtoks\@envbody
```

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

```
1117 \def\addto@envbody#1{\global\@envbody\@xp{\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.

```
1118 \def\collect@body#1{%
1119   \@envbody{\@xp#1\@xp{\the\@envbody}}%
1120   \edef\process@envbody{\the\@envbody\@nx\end{\@currenvir}}%
1121   \@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.

```
1122 \begingroup
1123 \@xp\let\curname\@currenvir\endcurname\collect@@body
```

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

```
1124 \edef\process@envbody{\@xp\@nx\curname\@currenvir\endcurname}%
1125 \process@envbody
1126 }
```

`\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.

```
1127 \def\push@begins#1\begin#2{%
1128   \ifx\end#2\else b\@xp\push@begins\fi
1129 }
```

`\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]

```
1130 \def\collect@@body#1\end#2{%
1131   \edef\begin@stack{\push@begins#1\begin\end \@xp\@gobble\begin@stack}%
1132   \ifx\@empty\begin@stack
1133     \endgroup
```

```

1134   \@checkend{#2}%
1135   \addto@envbody{#1}%
1136   \else
1137     \addto@envbody{#1\end{#2}}%
1138   \fi
1139   \process@envbody % A little tricky! Note the grouping
1140 }

```

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.

```

1141 \def\math@cr@@@aligned{%
1142   \ifodd\column@ \let\next@\@empty
1143   \else \def\next@{&\kern-\alignsep}%
1144   \fi
1145   \next@ \cr
1146 }

```

`\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.

```

1147 \newcommand{\start@aligned}[2]{%
1148   \RIfM@else
1149     \nonmatherr@{\begin{\@currenvir}}%
1150   \fi
1151   \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.

```

1152   \null\,%
1153   \if #1t\vtop \else \if#1b \vbox \else \vcenter \fi \fi \bgroup
1154     \maxfields@#2\relax
1155     \ifnum\maxfields@>\m@ne
1156       \multiply\maxfields@\tw@

```

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

```

1157     \let\math@cr@@@\math@cr@@@alignedat
1158     \alignsep@\z@skip
1159   \else
1160     \let\math@cr@@@\math@cr@@@aligned
1161     \alignsep@\minalignsep
1162   \fi

```

Reset the meaning of `\`.

```

1163   \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`.

```

1164   \default@tag
1165   \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.

```

1166     \global\column@\z@
1167     \ialign\bgroup
1168     &\column@plus
1169     \hfil
1170     \strut@
1171     $\m@th\displaystyle{##}$%
1172     \tabskip\z@skip
1173     &\column@plus
1174     $\m@th\displaystyle{{}##}$%
1175     \hfil
1176     \tabskip\alignsep@
1177     \crcr
1178 }

```

`\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 `\display@`, we also reset `\column@` here. Note than in `aligned`, `\column@` will increase without bound, since it never gets reset, but this is harmless.

```

1179 \def\math@cr@@@alignedat{%
1180   \ifnum\column@>\maxfields@
1181     \begingroup
1182     \measuring@false
1183     \@amsmath@err{Extra & on this line}%
1184     {\the\andhelp@}% "An extra & here is disastrous"
1185     \endgroup
1186     \fi
1187     \global\column@\z@
1188     \cr
1189 }

```

`\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.

```

1190 \def\alignsafe@testopt#1#2{%
1191   \relax\iffalse{\fi\ifnum' }=0\fi
1192   \@ifnextchar [%
1193     {\let\let@token\relax \ifnum'={\z@\fi\iffalse}\fi#1}%
1194     {\let\let@token\relax \ifnum'={\z@\fi\iffalse}\fi#1[#2]}%
1195 }

```

`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.

```

1196 \newenvironment{aligned}{%
1197   \let\@testopt\alignsafe@testopt
1198   \aligned@a
1199 }{%
1200   \crcr\egroup
1201   \restorecolumn@
1202   \egroup
1203 }
1204 \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}
```

```
1205 \newenvironment{alignedat}{%
1206   \let\@testopt\alignsafe@testopt
1207   \alignedat@a
1208 }{%
1209   \endaligned
1210 }
1211 \newcommand{\alignedat@a}[1][c]{\start@aligned{#1}}
```

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

```
1212 \newenvironment{gathered}[1][c]{%
1213   \RIfM@else
1214   \nonmatherr@{\begin{gathered}}%
1215   \fi
1216   \null\,%
1217   \if #1\vtop \else \if #1b\vbox \else \vcenter \fi\fi \bgroup
1218   \Let@ \chardef\dspbrk@context\@ne \restore@math@cr
1219   \spread@equation
1220   \ialign\bgroup
1221     \hfil\strut@$\m@th\displaystyle##$\hfil
1222     \crcr
1223 }{%
1224   \endaligned
1225 }
```

16.5 The gather environment

```
\start@gather
```

```
1226 \def\start@gather#1{%
1227   \RIfM@
1228   \nomath@env
1229   \DN@{\@namedef{end\@currenvir}{}\@gobble}%
1230   \else
1231     $$%
1232     #1%
1233     \ifst@rred \else \global\@eqnswtrue \fi
1234     \let\next@\gather@
1235   \fi
1236   \collect@body\next@
1237 }
```

```
gather
```

```
gather*
```

```
1238 \newenvironment{gather}{%
1239   \start@gather\st@rredfalse
1240 }{%
1241   \math@cr \black@\totwidth@ \egroup
1242   $$\ignorespacesafterend
1243 }
1244
1245 \newenvironment{gather*}{%
1246   \start@gather\st@rredtrue
1247 }{%
1248   \endgather
1249 }
```

```
\gather@
```

```
1250 \def\gather@#1{%
```



```

1251 \ingather@true \let\split\insplit@
1252 \let\tag\tag@in@align \let\label\label@in@display
1253 \chardef\dsprk@context\z@
1254 \intertext@ \disply@ \Let@
1255 \let\math@cr@@@math@cr@@gather
1256 \gmeasure@{#1}%
1257 \global\shifftag@false
1258 \tabskip\z@skip
1259 \global\row@\@ne
1260 \halign to\displaywidth\bgroup
1261 \strut@
1262 \setboxz@h{\m@th\displaystyle{##}$}%
1263 \calc@shift@gather
1264 \set@gather@field
1265 \tabskip\@centering
1266 &\setboxz@h{\strut@{##}}%
1267 \place@tag@gather
1268 \tabskip \iftagsleft@ \gdisplaywidth@ \else \z@skip \span\fi
1269 \crcr
1270 #1%
1271 }

```

\gmeasure@

```

1272 \def\gmeasure@#1{%
1273 \begingroup
1274 \measuring@true
1275 \totwidth@\z@
1276 \global\let\tag@lengths\@empty
1277 \savecounters@
1278 \setbox\@ne\vbox{%
1279 \everycr{\noalign{\global\tag@false
1280 \global\let\raise@tag\@empty \global\column@\z@}}%
1281 \let\label\@gobble
1282 \halign{%
1283 \setboxz@h{\m@th\displaystyle{##}$}%
1284 \ifdim\wdz@>\totwidth@
1285 \global\totwidth@\wdz@
1286 \fi
1287 &\setboxz@h{\strut@{##}}%
1288 \savetaglength@
1289 \crcr
1290 #1%
1291 \math@cr@@@
1292 }%
1293 }%
1294 \restorecounters@
1295 \if@fleqn
1296 \global\advance\totwidth@\@mathmargin
1297 \fi
1298 \iftagsleft@
1299 \ifdim\totwidth@>\displaywidth
1300 \global\let\gdisplaywidth@\totwidth@
1301 \else
1302 \global\let\gdisplaywidth@\displaywidth
1303 \fi
1304 \fi
1305 \endgroup
1306 }

```

\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.

```

1307 \def\math@cr@@@gather{%
1308   \ifst@rred\nonumber\fi
1309   &\relax
1310   \make@display@tag
1311   \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.

```

1312   \global\advance\row@\@ne
1313   \cr
1314 }

```

`\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 T_EX'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.

```

1315 \def\calc@shift@gather{%
1316   \dimen@\mintagsep\relax
1317   \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.

```

1318   \if@fleqn
1319     \global\eqnshift@\@mathmargin
1320     \ifdim\tagwidth@>\z@
1321       \advance\dimen@\tagwidth@
1322       \iftagsleft@
1323         \ifdim\dimen@>\@mathmargin
1324           \global\shifftag@true
1325         \fi
1326       \else
1327         \advance\dimen@\@mathmargin
1328         \advance\dimen@\wdz@
1329         \ifdim\dimen@>\displaywidth
1330           \global\shifftag@true
1331         \fi
1332       \fi
1333     \fi
1334   \else
1335     \global\eqnshift@\displaywidth
1336     \global\advance\eqnshift@-\wdz@
1337     \ifdim\tagwidth@>\z@
1338       \multiply\dimen@\tw@
1339       \advance\dimen@\wdz@
1340       \advance\dimen@\tagwidth@
1341       \ifdim\dimen@>\displaywidth
1342         \global\shifftag@true
1343       \else

```

```

1344         \ifdim\eqnshift@<4\tagwidth@
1345         \global\advance\eqnshift@-\tagwidth@
1346         \fi
1347     \fi
1348 \fi
1349 \global\divide\eqnshift@\tw@
1350 \iftagsleft@
1351     \global\eqnshift@-\eqnshift@
1352     \global\advance\eqnshift@\displaywidth
1353     \global\advance\eqnshift@-\wdz@
1354 \fi
1355 \ifdim\eqnshift@<\z@
1356     \global\eqnshift@\z@
1357 \fi
1358 \fi
1359 }

```

`\place@tag@gather`

`\set@gather@field`

```

1360 \def\place@tag@gather{%
1361     \iftagsleft@
1362         \kern-\gdisplaywidth@
1363         \ifshifftag@
1364             \rlap{\vbox{%
1365                 \normalbaselines
1366                 \boxz@
1367                 \vbox to\lineht@{}}%
1368             \raise@tag
1369         }}%
1370     \global\shifftag@false
1371 \else
1372     \rlap{\boxz@}%
1373 \fi
1374 \else
1375     \ifdim\totwidth@>\displaywidth
1376         \dimen@\totwidth@
1377         \advance\dimen@-\displaywidth
1378         \kern-\dimen@
1379     \fi
1380     \ifshifftag@
1381         \llap{\vtop{%
1382             \raise@tag
1383             \normalbaselines
1384             \setbox@ne\@null
1385             \dp\@ne\lineht@
1386             \box\@ne
1387             \boxz@
1388         }}%
1389     \global\shifftag@false
1390 \else
1391     \llap{\boxz@}%
1392 \fi
1393 \fi
1394 }
1395 %
1396 \def\set@gather@field{%
1397     \iftagsleft@
1398         \global\lineht@\ht\z@
1399 \else
1400     \global\lineht@\dp\z@

```

```

1401 \fi
1402 \kern\eqnshift@
1403 \boxz@
1404 \hfil
1405 }

```

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@
1406 \newif\ifxxat@
1407
1408 \newif\ifcheckat@
1409
1410 \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.

```

1411 \def\start@align#1#2#3{%
1412   \let\xatlevel@#1% always \z@, \@ne, or \tw@
1413   \maxfields@#3\relax
1414   \ifnum\maxfields@>\m@ne
1415     \checkat@true
1416     \ifnum\xatlevel@=\tw@
1417       \xxat@true
1418     \fi
1419     \multiply\maxfields@\tw@
1420   \else
1421     \checkat@false
1422   \fi
1423   \ifingather@
1424     \iffalse{\fi\ifnum0=} \fi
1425     \DN@{\vcenter\bgroup\savealignstate@\align@#2}%
1426   \else
1427     \ifmode
1428       \if@display
1429         \DN@{\align@recover}%
1430       \else
1431         \nomath@env
1432         \DN@{\@namedef{end\@currenenv}{}\@gobble}%
1433       \fi
1434     \else
1435       $$%
1436       \let\split\insplit@
1437       \DN@{\align@#2}%
1438     \fi

```

```

1439   \fi
1440   \collect@body\next@
1441 }

```

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.

```

1442 \def\align@recover#1#2#3{%
1443   \endgroup
1444   \@amsmath@err{%
1445     Erroneous nesting of equation structures;\MessageBreak
1446     trying to recover with 'aligned'%
1447   }\@ehc
1448   \begin{aligned}\relax#1\end{aligned}%
1449 }

```

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

```

align* 1450 \newenvironment{alignat}{%
flalign 1451   \start@align\z@\st@rredfalse
flalign* 1452 }{%
alignat 1453   \endalign
alignat* 1454 }
xalignat 1455 \newenvironment{alignat*}{%
xalignat* 1456   \start@align\z@\st@rredtrue
xxalignat 1457 }{%
1458   \endalign
1459 }
1460 \newenvironment{xalignat}{%
1461   \start@align\@ne\st@rredfalse
1462 }{%
1463   \endalign
1464 }
1465 \newenvironment{xalignat*}{%
1466   \start@align\@ne\st@rredtrue
1467 }{%
1468   \endalign
1469 }
1470 \newenvironment{xxalignat}{%
1471   \start@align\tw@\st@rredtrue
1472 }{%
1473   \endalign
1474 }
1475 \newenvironment{align}{%
1476   \start@align\@ne\st@rredfalse\m@ne
1477 }{%
1478   \math@cr \black@\totwidth@
1479   \egroup
1480   \ifingather@
1481   \restorealignstate@

```

```

1482   \egroup
1483   \nonumber
1484   \ifnum0='{\fi\iffalse}\fi
1485   \else
1486     $$%
1487   \fi
1488   \ignorespacesafterend
1489 }
1490 \newenvironment{align*}{%
1491   \start@align\@ne\st@rredtrue\m@ne
1492 }{%
1493   \endalign
1494 }
1495 \newenvironment{flalign}{%
1496   \start@align\tw@\st@rredfalse\m@ne
1497 }{%
1498   \endalign
1499 }
1500 \newenvironment{flalign*}{%
1501   \start@align\tw@\st@rredtrue\m@ne
1502 }{%
1503   \endalign
1504 }

```

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

```

1505 \def\align@#1#2{%
1506   \inalign@true \intertext@ \Let@ \chardef\dspbrk@context\z@
1507   \ifingather@\else\display@\fi
1508   \let\math@cr@@@\math@cr@@@align
1509   \ifxat@\else \let\tag\tag@in@align \fi
1510   \let\label\label@in@display
1511   #1% set st@r
1512   \ifst@rred\else \global\@eqnswtrue \fi
1513   \measure@{#2}%
1514   \global\row@\z@
1515   \tabskip\eqnshift@
1516   \halign\bgroup
1517     \span\align@preamble\crrc
1518     #2%
1519 }

```

`\math@cr@@@align`

```

1520 \def\math@cr@@@align{%
1521   \ifst@rred\nonumber\fi
1522   \if@eqnsw \global\tag@true \fi
1523   \global\advance\row@\@ne
1524   \add@amps\maxfields@
1525   \omit
1526   \kern-\alignsep@
1527   \iftag@
1528     \setboxz@h{\@lign\strut@{\make@display@tag}}%
1529     \place@tag
1530   \fi
1531   \ifst@rred\else\global\@eqnswtrue\fi
1532   \global\lineht@\z@
1533   \cr
1534 }

```

`\cr@@@align@measure`

```

1535 \def\math@cr@@@align@measure{%
1536   &\omit
1537   \global\advance\row@\@ne
1538   \ifst@rred\nonumber\fi
1539   \if@eqnsw \global\tag@true \fi
1540   \ifnum\column@>\maxfields@
1541     \ifcheckat@
1542       \begingroup
1543         \measuring@false
1544         \@amsmath@err{Extra & on this line}%
1545         {\the\andhelp@}% "An extra & here is disastrous"
1546       \endgroup
1547     \else
1548       \global\maxfields@\column@
1549     \fi
1550   \fi
1551   \setboxz@h{\@lign\strut@{%
1552     \if@eqnsw
1553       \stepcounter{equation}%
1554       \tagform@\theequation
1555     \else
1556       \iftag@\df@tag\fi
1557     \fi
1558   }}%
1559   \savetaglength@
1560   \ifst@rred\else\global\@eqnswtrue\fi
1561   \cr
1562 }
```

`\field@lengths`

```

\savefieldlength@ 1563 \let\field@lengths\@empty
\fieldlengths@    1564
1565 \def\savefieldlength@{%
1566   \begingroup
1567     \let\or\relax
1568     \xdef\field@lengths{%
1569       \field@lengths
1570       \ifnum\column@=0
1571         \or
1572       \else
1573         ,%
1574       \fi
1575       \the\wdz@
1576     }%
1577   \endgroup
1578 }
1579
1580 \def\fieldlengths@#1{%
1581   \ifcase\@xp#1\field@lengths\fi
1582 }
```

`\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`.

```
1583 \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 `\dimen`, 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.

```
1584 \def\maxcol@width#1{%
1585   \ifcase\@xp#1\maxcolumn@widths\fi\relax
1586 }
```

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`:

$$\begin{array}{cccccc}
 1 & 2 & 3 & 4 & 5 & 6 \\
 \left| V_i + q_i v_j \right| = v_i, & & \left| X_i \right| = x_i - q_i x_j, & & \left| U_i \right| = u_i, & \text{for } i \neq j; \\
 \left| V_j \right| = v_j, & & \left| X_j \right| = x_j, & & \left| U_j \right| = u_j + \sum_{i \neq j} q_i u_i. & \\
 & & & & & (4)
 \end{array} \quad (3)$$

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.

```

1587 \def\measure@#1{%
1588     \begingroup
1589         \measuring@true
1590         \global\eqnshift@z@
1591         \global\alignsep@z@
1592         \global\let\tag@lengths\@empty
1593         \global\let\field@lengths\@empty
1594         \savecounters@
1595         \global\setbox0\vbox{%
1596             \let\math@cr@@@math@cr@@@align@measure
1597             \everycr{\noalign{\global\tag@false
1598                 \global\let\raise@tag\@empty \global\column@z@}}%
1599             \let\label\@gobble
1600             \global\row@z@
1601             \tabskipz@
1602             \halign{\span\align@preamble\crr
1603                 #1%
1604                 \math@cr@@@
1605                 \global\column@z@
1606                 \add@amps\maxfields@\cr
1607             }%
1608         }%
1609     \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.

```

1610     \ifodd\maxfields@
1611         \global\advance\maxfields@\@ne
1612     \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?

```

1613     \ifnum\xatlevel@=\tw@
1614         \ifnum\maxfields@<\thr@@
1615             \let\xatlevel@z@
1616         \fi
1617     \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.)

```
1618     \setbox\z@\vbox{%
1619         \unvbox\z@ \unpenalty \global\setbox\@ne\lastbox
1620     }%
```

`\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):

```
1621     \global\totwidth@\wd\@ne
1622     \iffleqn \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.

```
1623     \global\let\maxcolumn@widths\@empty
1624     \begingroup
1625         \let\or\relax
1626         \loop
1627             \global\setbox\@ne\hbox{%
1628                 \unhbox\@ne \unskip \global\setbox\thr@@\lastbox
1629             }%
1630         \ifhbox\thr@@
1631             \xdef\maxcolumn@widths{ \or \the\wd\thr@@ \maxcolumn@widths}%
1632         \repeat
1633     \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.

```
1634     \dimen@\displaywidth
1635     \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.

```
1636     \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`.

```

1637      \global\alignsep@\z@
1638      \let\minalignsep\z@
1639      \@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.

```

1640      \if@fleqn
1641          \@tempcnta\@ne
1642          \global\eqnshift@\@mathmargin
1643      \else
1644          \@tempcnta\tw@
1645          \global\eqnshift@\dimen@
1646          \global\divide\eqnshift@\@tempcnta
1647      \fi
1648      \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.

```

1649      \@tempcntb\maxfields@
1650      \divide\@tempcntb\tw@
1651      \@tempcnta\@tempcntb
1652      \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.

```

1653      \if@fleqn
1654          \global\eqnshift@\@mathmargin
1655          \global\alignsep@\dimen@
1656          \global\divide\alignsep@\@tempcnta
1657      \else

```

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

```

1658          \global\advance\@tempcnta\@ne
1659          \global\eqnshift@\dimen@
1660          \global\divide\eqnshift@\@tempcnta
1661          \global\alignsep@\eqnshift@
1662      \fi
1663      \or

```

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

```

1664      \@tempcntb\maxfields@
1665      \divide\@tempcntb\tw@
1666      \global\advance\@tempcntb\m@ne
1667      \global\@tempcnta\@tempcntb
1668      \global\eqnshift@\z@
1669      \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.

```

1670      \if@fleqn
1671          \global\advance\alignsep@\@mathmargin\relax
1672      \fi
1673      \global\divide\alignsep@\@tempcntb
1674      \fi

```

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

```

1675      \ifdim\alignsep@<\minalignsep\relax
1676          \global\alignsep@\minalignsep\relax

```

```

1677         \ifdim\eqnshift@>\z@
1678         \if@fleqn\else
1679             \global\eqnshift@\displaywidth
1680             \global\advance\eqnshift@-\totwidth@
1681             \global\advance\eqnshift@-\@tempcntb\alignsep@
1682             \global\divide\eqnshift@\tw@
1683         \fi
1684     \fi
1685 \fi
1686 \ifdim\eqnshift@<\z@
1687     \global\eqnshift@\z@
1688 \fi
1689 \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.

```

1690     \global\tagshift@\totwidth@
1691     \global\advance\tagshift@\@tempcntb\alignsep@
1692 \if@fleqn
1693     \ifnum\xatlevel@=\tw@
1694         \global\advance\tagshift@-\@mathmargin\relax
1695     \fi
1696 \else
1697     \global\advance\tagshift@\eqnshift@
1698 \fi
1699 \iftagsleft@ \else
1700     \global\advance\tagshift@-\displaywidth
1701 \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 `\block`) at the end of the environment.

```

1702     \dimen@\minalignsep\relax
1703     \global\advance\totwidth@\@tempcntb\dimen@
1704     \ifdim\totwidth@>\displaywidth
1705         \global\let\displaywidth@\totwidth@
1706     \else
1707         \global\let\displaywidth@\displaywidth
1708     \fi
1709 \endgroup
1710 }

```

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`.

```
1711 \iftagsleft@\if@fleqn
1712     \def\calc@shift@alignf%
1713         \global\let\tag@shifts\@empty
1714         \begingroup
```

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

```
1715         \@tempdima\@mathmargin\relax
1716         \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.

```
1717         \loop
1718             \ifnum\row@>0
1719                 \ifdim\tag@width\row@>\z@
1720                     \x@calc@shift@lf
1721                 \else
1722                     \saveshift@0%
1723                 \fi
1724                 \advance\row@\@m@ne
1725             \repeat
1726         \endgroup
1727     }
```

`\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.

```
1728     \def\x@calc@shift@lf%
1729         \ifdim\eqnshift@=\z@
1730             \global\eqnshift@\@mathmargin\relax
1731             \alignsep@\displaywidth
1732             \advance\alignsep@-\totwidth@
1733             \global\divide\alignsep@\@tempcntb
1734             \ifdim\alignsep@<\minalignsep\relax
1735                 \global\alignsep@\minalignsep\relax
1736             \fi
1737         \fi
1738         \ifdim\tag@width\row@>\@tempdima
1739             \saveshift@1%
1740         \else
1741             \saveshift@0%
1742         \fi
1743     }
1744 \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 separatin. 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`.

```

1745 \iftagsleft@\else\if@fleqn
1746   \def\calc@shift@align{%
1747     \global\let\tag@shifts\@empty
1748     \begingroup
1749     \loop
1750       \ifnum\row@>0
1751         \ifdim\tag@width\row@>\z@
1752           \x@calc@shift@rf
1753         \else
1754           \saveshift@0%
1755         \fi
1756         \advance\row@\m@ne
1757     \repeat
1758   \endgroup
1759 }
```

`\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.

```

1760   \def\x@calc@shift@rf{%
1761     \column@\z@
1762     \@tempdimb\z@
1763     \@tempdimc\z@
1764     \edef\@tempb{\fieldlengths@\row@}%
1765     \@for\@tempa:=\@tempb\do{%
1766       \advance\column@\@ne
1767       \x@rcalc@width
1768     }%
1769   \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.

```

1770     \advance\column@\m@ne
1771     \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@`.

```

1772     \ifnum\@tempcntb>\column@
1773       \advance\@tempcnta-\@tempcntb
1774       \advance\@tempcnta\column@
1775     \@tempcntb\column@
1776   \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.

```

1777         \tagwidth@\tag@width\row@\relax
1778         \@tempdima\eqnshift@
1779         \advance\@tempdima\@tempdimc\relax
1780         \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.

```

1781         \dimen@\minalignsep\relax
1782         \multiply\dimen@\@tempcntb
1783         \advance\dimen@\mintagsep\relax
1784         \advance\dimen@\@tempdima
1785         \ifdim\dimen@>\displaywidth
1786             \saveshift@1%
1787         \else
1788             \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.

```

1789         \dimen@\alignsep@\relax
1790         \multiply\dimen@\@tempcntb
1791         \advance\dimen@\@tempdima
1792         \advance\dimen@\tagwidth@
1793         \ifdim\dimen@>\displaywidth
1794             \dimen@\displaywidth
1795             \advance\dimen@-\@tempdima
1796             \ifnum\xatlevel@=\tw@
1797                 \advance\dimen@-\mintagsep\relax
1798             \fi
1799             \divide\dimen@\@tempcnta
1800             \ifdim\dimen@<\minalignsep\relax
1801                 \global\alignsep@\minalignsep\relax
1802             \else
1803                 \global\alignsep@\dimen@
1804             \fi
1805         \fi
1806     \fi
1807 \endgroup
1808 }
1809 \fi\fi

```

Tag placement when `\tagsleft@false`, `\@fleqn@false`. 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
```

```
1810 \iftagsleft@\else\if@fleqn\else
```

```

1811 \def\calc@shift@align{%
1812   \global\let\tag@shifts\@empty
1813   \begingroup
1814     \loop
1815       \ifnum\row@>0
1816         \ifdim\tag@width\row@>\z@
1817           \x@calc@shift@rc
1818         \else
1819           \saveshift@0%
1820         \fi
1821       \advance\row@\m@ne
1822     \repeat
1823   \endgroup
1824 }

```

\x@calc@shift@rc

```

1825 \def\x@calc@shift@rc{%
1826   \column@\z@
1827   \@tempdimb\z@
1828   \@tempdimc\z@
1829   \edef\@tempb{\fieldlengths@\row@}%
1830   \@for\@tempa:=\@tempb\do{%
1831     \advance\column@\@ne
1832     \x@rcalc@width
1833   }%
1834   \begingroup
1835     \advance\column@\m@ne
1836     \divide\column@\tw@
1837     \ifnum\@tempcntb>\column@
1838       \advance\@tempcnta-\@tempcntb
1839       \advance\@tempcnta\column@
1840       \@tempcntb\column@
1841     \fi
1842     \tagwidth@\tag@width\row@\relax
1843     \@tempdima\@tempdimc
1844     \advance\@tempdima\tagwidth@
1845     \dimen@\minalignsep\relax
1846     \multiply\dimen@\@tempcntb
1847     \advance\dimen@\mintagsep\relax
1848     \ifnum\xatlevel@=\tw@ \else
1849       \advance\dimen@\mintagsep\relax
1850     \fi
1851     \advance\dimen@\@tempdima
1852     \ifdim\dimen@>\displaywidth
1853       \saveshift@1%
1854     \else
1855       \saveshift@0%
1856       \dimen@\eqnshift@
1857       \advance\dimen@\@tempdima
1858       \advance\dimen@\@tempcntb\alignsep@
1859       \advance\dimen@\tagwidth@
1860       \ifdim\dimen@>\displaywidth
1861         \dimen@\displaywidth
1862         \advance\dimen@-\@tempdima
1863         \ifnum\xatlevel@=\tw@
1864           \advance\dimen@-\mintagsep\relax
1865         \fi
1866       \divide\dimen@\@tempcnta
1867       \ifdim\dimen@<\minalignsep\relax

```



```

1868             \global\alignsep@\minalignsep\relax
1869             \eqnshift@\displaywidth
1870             \advance\eqnshift@-\@tempdima
1871             \advance\eqnshift@-\@tempcntb\alignsep@
1872             \global\divide\eqnshift@\tw@
1873         \else
1874             \ifdim\dimen@<\eqnshift@
1875                 \ifdim\dimen@<\z@
1876                     \global\eqnshift@\z@
1877                 \else
1878                     \global\eqnshift@\dimen@
1879                 \fi
1880             \fi
1881             \ifdim\dimen@<\alignsep@
1882                 \global\alignsep@\dimen@
1883             \fi
1884         \fi
1885     \fi
1886 \fi
1887 \endgroup
1888 }
1889 \fi\fi

```

`\x@rcalc@width`

```

1890 \iftagsleft@\else
1891     \def\x@rcalc@width{%
1892         \ifdim\@tempa > \z@
1893             \advance\@tempdimc\@tempdimb
1894             \ifodd\column@
1895                 \advance\@tempdimc\maxcol@width\column@
1896                 \@tempdimb\z@
1897             \else
1898                 \advance\@tempdimc\@tempa\relax
1899                 \@tempdimb\maxcol@width\column@
1900                 \advance\@tempdimb-\@tempa\relax
1901             \fi
1902         \else
1903             \advance\@tempdimb\maxcol@width\column@\relax
1904         \fi
1905     }
1906 \fi

```

Tag placement when `\tagsleft@true`, `\@fleqn@false`.

`\calc@shift@align`

```

1907 \iftagsleft@\if@fleqn\else
1908     \def\calc@shift@align{%
1909         \global\let\tag@shifts\@empty
1910         \begingroup
1911             \loop
1912                 \ifnum\row@>\z@
1913                     \ifdim\tag@width\row@>\z@
1914                         \x@calc@shift@lc
1915                     \else
1916                         \saveshift@0%
1917                     \fi
1918                     \advance\row@\m@ne
1919                 \repeat
1920         \endgroup

```

1921 }
 1922

`\x@calc@shift@lc`

1922 `\def\x@calc@shift@lc{%`
 1923 `\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.

1924 `\@tempdima\z@ % ‘width of equation’`
 1925 `\@tempdimb\z@ % ‘indent of equation’`
 1926 `\edef\@tempb{\fieldlengths@\row@}%`
 1927 `\@for\@tempa:=\@tempb\do{%`
 1928 `\advance\column@\@ne`
 1929 `\x@lcalc@width`
 1930 `}%`
 1931 `\begingroup`
 1932 `\tagwidth@\tag@width\row@\relax`

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

1933 `\@tempdima\totwidth@`
 1934 `\advance\@tempdima-\@tempdimb`
 1935 `\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.

1936 `\dimen@\minalignsep\relax`
 1937 `\multiply\dimen@\@tempcntb`
 1938 `\advance\dimen@\mintagsep\relax`
 1939 `\ifnum\xatlevel@=\tw@ \else`
 1940 `\advance\dimen@\mintagsep\relax`
 1941 `\fi`
 1942 `\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.

1943 `\ifdim\dimen@>\displaywidth`
 1944 `\saveshift@1%`
 1945 `\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.

1946 `\saveshift@0%`
 1947 `\dimen@\alignsep@`
 1948 `\multiply\dimen@\count@`
 1949 `\advance\dimen@\eqnshift@`
 1950 `\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.

```

1951         \ifdim\dimen@<2\tagwidth@
1952             \dimen@\displaywidth
1953             \advance\dimen@-\@tempdima
1954             \ifnum\xatlevel@=\tw@
1955                 \advance\dimen@-\mintagsep\relax
1956         \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].

```

1957         \ifnum\@tempcnta>\z@
1958             \divide\dimen@\@tempcnta
1959         \else \dimen@\z@
1960         \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.

```

1961         \ifdim\dimen@<\minalignsep\relax
1962             \global\alignsep@\minalignsep\relax
1963             \dimen@\displaywidth
1964             \advance\dimen@-\@tempdima
1965             \advance\dimen@-\@tempcntb\alignsep@
1966             \global\divide\dimen@\tw@
1967         \else
1968             \ifdim\dimen@<\alignsep@
1969                 \global\alignsep@\dimen@
1970             \fi
1971         \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.

```

1972         \ifnum\xatlevel@=\tw@
1973             \dimen@\mintagsep\relax
1974         \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.

```

1975         \advance\dimen@\tagwidth@
1976         \advance\dimen@-\@tempdimb
1977         \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.

```

1978         \ifdim\dimen@>\eqnshift@
1979             \global\eqnshift@\dimen@
1980         \fi

```

```

1981         \fi
1982     \fi
1983     \endgroup
1984 }

```

`\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.

```

1985     \def\x@lcalc@width{%
1986         \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.

```

1987         \ifdim\@tempa > \z@
1988             \@tempdima\p@
1989             \ifodd\column@
1990                 \advance\@tempdimb \maxcol@width\column@
1991                 \advance\@tempdimb-\@tempa
1992             \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.

```

1993         \count@\column@
1994         \advance\count@\m@ne
1995         \divide\count@\tw@
1996         \advance\@tempcnta-\count@
1997         \advance\@tempcntb-\count@
1998     \else
1999         \advance\@tempdimb \maxcol@width\column@\relax
2000     \fi
2001 \fi
2002 }
2003 \fi\fi

```

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

```

2004 \def\place@tag{%
2005     \iftagsleft@
2006         \kern-\tagshift@
2007         \if1\shift@tag\row@\relax
2008             \rlap{\vbox{%
2009                 \normalbaselines
2010                 \boxz@
2011                 \vbox to\lineht@{}}%
2012             \raise@tag
2013         }}%
2014     \else

```

```

2015         \rlap{\boxz@}%
2016     \fi
2017     \kern\displaywidth@
2018 \else
2019     \kern-\tagshift@
2020     \if1\shift@tag\row@\relax

```

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

```

2021         \llap{\vtop{%
2022             \raise@tag
2023             \normalbaselines
2024             \setbox\@ne\null
2025             \dp\@ne\lineht@
2026             \box\@ne
2027             \boxz@
2028         }}%
2029 \else
2030     \llap{\boxz@}%
2031 \fi
2032 \fi
2033 }

```

`\align@preamble`

```

2034 \def\align@preamble{%
2035     &\hfil
2036     \strut@
2037     \setboxz@h{\@lign$\m@th\displaystyle{##}$}%
2038     \ifmeasuring@\savefieldlength@\fi
2039     \set@field
2040     \tabskip\z@skip
2041     &\setboxz@h{\@lign$\m@th\displaystyle{ }##}$}%
2042     \ifmeasuring@\savefieldlength@\fi
2043     \set@field
2044     \hfil
2045     \tabskip\alignsep@
2046 }

```

`\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.

```

2047 \def\set@field{%
2048     \column@plus
2049     \iftagsleft@
2050         \ifdim\ht\z@>\lineht@
2051             \global\lineht@\ht\z@
2052         \fi
2053     \else
2054         \ifdim\dp\z@>\lineht@
2055             \global\lineht@\dp\z@
2056         \fi
2057     \fi
2058     \boxz@
2059 }

```

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.

```

2060 \edef\split@err#1{%
2061     \@nx\@amsmath@err{%
2062         \string\begin{split} won't work here%

```

```

2063   }{%
2064     \@xp\@nx\csname
2065     Did you forget a preceding \string\begin{equation}?^^J%
2066     If not, perhaps the 'aligned' environment is what
2067     you want.\endcsname}%
2068 }

```

`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.

```

2069 \newenvironment{split}{%
2070   \if@display
2071     \ifinner
2072       \@xp\@xp\@xp\split@aligned
2073     \else
2074       \ifst@rred \else \global\@eqnswtrue \fi
2075     \fi
2076   \else \let\endsplit\@empty \@xp\collect@body\@xp\split@err
2077   \fi
2078   \collect@body\gather@split
2079 }{%
2080   \crr
2081   \egroup
2082   \egroup
2083   \iftagsleft@ \@xp\lendsplit@ \else \@xp\rendsplitt@ \fi
2084 }

2085 \let\split@tag\relax % init

2086 \def\gather@split#1#2#3{%
2087   \@xp\endgroup \reset@equation % math@cr will handle equation numbering
2088   \iftag@
2089     \toks@\@xp{\df@tag}%
2090     \edef\split@tag{%
2091       \gdef\@nx\df@tag{\the\toks@}%
2092       \global\@nx\tag@true \@nx\nonumber
2093     }%
2094   \else \let\split@tag\@empty
2095   \fi
2096   \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]

```

2097   \vcenter\bgroup
2098     \gather@\split@tag \begin{split}#1\end{split}}%
2099   \def\endmathdisplay@a{%
2100     \math@cr \black@ \totwidth@ \egroup
2101     \egroup
2102   }%
2103 }

```

`\insplit@`

```

2104 \def\insplit@{%
2105   \global\setbox\z@\vbox\bgroup
2106   \let@ \chardef\dsprk@context\@ne \restore@math@cr
2107   \default@tag % disallow use of \tag here
2108   \ialign\bgroup
2109     \hfil

```

```

2110     \strut@
2111     $\m@th\displaystyle{##}$%
2112     &$\m@th\displaystyle{{}##}$%
2113     \hfill % Why not \hfil?---dmj, 1994/12/28
2114     \crrc
2115 }

```

`\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`?

```

2116 \def\rendsplit@{%
2117     \ifinalign@

```

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

```

2118         \global\setbox9 \vtop{%
2119             \unvcopy\z@
2120             \global\setbox8 \lastbox
2121             \unskip
2122         }%
2123         \setbox\@ne\hbox{%
2124             \unhcopy8
2125             \unskip
2126             \global\setbox\tw@\lastbox
2127             \unskip
2128             \global\setbox\thr@@\lastbox
2129         }%
2130         \ifctagsplit@
2131         \gdef\split@{%
2132             \hbox to\wd\thr@@{%
2133                 &\vcenter{\vbox{\moveleft\wd\thr@@\boxz@}}%
2134             }%
2135         \else
2136         \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.

```

2137         \gdef\split@{%
2138             \global\@tempcnta\column@
2139             &\setboxz@h{%
2140                 \savetaglength@
2141                 \global\advance\row@\@ne
2142                 \vbox{\moveleft\wd\thr@@\box9}%
2143                 \crrc
2144                 \noalign{\global\lineht@\z@}%
2145                 \add@amps\@tempcnta
2146                 \box\thr@@
2147                 &\box7
2148             }%
2149         \fi

```

```

2150   \else
2151     \ifctagsplit@
2152     \gdef\split@{\vcenter{\boxz@}}%
2153     \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

```

2154     \gdef\split@{%
2155       \boxz@
2156 %     \box9
2157 %     \crrc
2158 %     \hbox{\box\thr@@\box7}%
2159     }%
2160   \fi
2161 \fi
2162 \aftergroup\split@
2163 }

```

`\lendsplit@`

```

2164 \def\lendsplit@{%
2165   \global\setbox9\vtop{\unvcopy\z@}%
2166   \ifinalign@

```

Moved following two boxes inside the `\ifinalign@`, 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

```

2167   \setbox\@ne\vbox{%
2168     \unvcopy\z@
2169     \global\setbox8\lastbox
2170   }%
2171   \setbox\@ne\hbox{%
2172     \unhcopy8%
2173     \unskip
2174     \setbox\tw@\lastbox
2175     \unskip
2176     \global\setbox\thr@@\lastbox
2177   }%
2178   \ifctagsplit@
2179     \gdef\split@{%
2180       \hbox to\wd\thr@@{%
2181         &\vcenter{\vbox{\moveleft\wd\thr@@\box9}}%
2182       }%
2183   \else
2184     \gdef\split@{%
2185       \hbox to\wd\thr@@{%
2186         &\vbox{\moveleft\wd\thr@@\box9}%
2187       }%
2188   \fi
2189 \else
2190   \ifctagsplit@
2191     \gdef\split@{\vcenter{\box9}}%
2192   \else
2193     \gdef\split@{\box9}%
2194   \fi
2195 \fi
2196 \aftergroup\split@
2197 }

```


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`.

```
2198 \def\split@aligned#1#2{%
2199   \iffalse{\fi\ifnum0='}\fi
2200   \collect@body\split@al@a}

2201 \def\split@al@a#1#2#3{%
2202   \split@warning
2203   \endgroup
```

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

```
2204 \toks@{\begin{aligned}}%
2205 \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.

```
2206 \the\toks@\relax#1\end{aligned}%
2207 \ifnum0='{ \fi\iffalse}\fi
2208 }
```

```
2209 \def\split@al@tagcheck{%
2210   \ifctagsplit@
2211   \else
2212     \iftagsleft@ \toks@\@xp{\the\toks@ [t]}%
2213     \else \toks@\@xp{\the\toks@ [b]}%
2214     \fi
2215   \fi
2216 }
```

```
2217 \def\split@warning{%
2218   \PackageWarning{amsmath}{%
2219     Cannot use 'split' here;\MessageBreak trying to recover with 'aligned'}%
2220 }
```

16.8 The multiline environment

In the original $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\mathcal{T}\mathcal{E}\mathcal{X}$, `\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@`.

```
2221 \newskip\multilinegap
2222 \multilinegap10pt
2223 \newskip\multlinetaggap
2224 \multlinetaggap10pt
```

`\start@multiline`

```
2225 \def\start@multiline#1{%
2226   \RIfM@
2227   \nomath@env
2228   \DN@{\@namedef{end\@currenenv}{}\@gobble}%
2229   \else
2230     $$%
2231     #1%
2232     \ifst@rred
```

```

2233         \nonumber
2234     \else
2235         \global\@eqnswtrue
2236     \fi
2237     \let\next@\multline@
2238 \fi
2239 \collect@body\next@
2240 }

multiline
multiline*
2241 \newenvironment{multiline}{%
2242 \start@multiline\st@rredfalse
2243 }{%
2244 \iftagsleft@ \exp\lendmultiline@ \else \exp\rendmultiline@ \fi
2245 \ignorespacesafterend
2246 }

2247 \newenvironment{multiline*}{\start@multiline\st@rredtrue}{\endmultiline}

\multline@
2248 \def\multline@#1{%
2249     \Let@

For multiline neither \display@ 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.)

2250     \@display@init{\global\advance\row@\@ne \global\dspbrk@lvl\m@ne}%
2251     \chardef\dspbrk@context\z@
2252     \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.

2253     \let\tag\tag@in@align
2254     \global\tag@false \global\let\raise@tag\@empty
2255     \mmeasure@{#1}%
2256     \let\tag@gobble\tag \let\label\@gobble
2257     \tabskip \if@fleqn \@mathmargin \else \z@skip \fi
2258     \totwidth@\displaywidth
2259     \if@fleqn
2260         \advance\totwidth@-\@mathmargin
2261     \fi
2262     \halign\bgroup
2263         \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 \lendmultiline@.

2264         \if@fleqn
2265             \hskip \@centering \relax
2266         \else
2267             \hfil
2268         \fi
2269         \strut@
2270         $\m@th\displaystyle{ }##\endmultiline@math
2271         \hfil
2272     }%

```

2273 \crr

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.

```

2274           \if@fleqn
2275               \hskip-\@mathmargin
2276           \def\multline@indent{\hskip\@mathmargin}% put it back
2277       \else
2278           \hfilneg
2279           \def\multline@indent{\hskip\multlinegap}%
2280       \fi
2281       \iftagsleft@
2282           \iftag@
2283               \begingroup
2284                   \ifshifftag@
2285                       \rlap{\vbox{%
2286                            \normalbaselines
2287                            \hbox{%
2288                                \strut@
2289                                \make@display@tag
2290                            }%
2291                        \vbox to\lineht@{%
2292                            \raise@tag
2293                        }}%

```

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@.

```

2294                   \multline@indent
2295           \else
2296               \setbox\z@\hbox{\make@display@tag}%
2297               \dimen@\@mathmargin \advance\dimen@-\wd\z@
2298               \ifdim\dimen@<\multlinetaggap
2299                   \dimen@\multlinetaggap
2300               \fi
2301               \box\z@ \hskip\dimen@\relax
2302           \fi
2303       \endgroup
2304       \else
2305           \multline@indent
2306       \fi
2307       \else
2308           \multline@indent
2309       \fi
2310       #1%
2311 }

```

An extra level of indirection for the closing \$ in multiline 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.

```
2312 \def\endmultline@math{}
```

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

```
2313 \def\lendmultline@{%
2314     \hfilneg
2315     \hskip\multlinegap
2316     \math@cr
2317     \egroup
2318     $$$%
2319 }
```

`\rendmultline@`

```
2320 \def\rendmultline@{%
2321     \iftag@
2322     $\let\endmultline@math\relax
2323     \ifshifftag@
2324     \hskip\multlinegap
```

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

```
2325         \llap{\vtop{%
2326             \raise@tag
2327             \normalbaselines
2328             \setbox\@ne\null
2329             \dp\@ne\lineht@
2330             \box\@ne
2331             \hbox{\strut@make@display@tag}}%
2332     }}%
2333     \else
2334         \hskip\multlinetaggap
2335         \make@display@tag
2336     \fi
2337 \else
2338     \hskip\multlinegap
2339 \fi
2340 \hfilneg
```

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

```
2341     \math@cr
2342     \egroup$$$%
2343 }
```

`\mmeasure@`

```
2344 \def\mmeasure@#1{%
2345     \begingroup
2346     \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]

```
2347     \def\label##1{%
2348         \begingroup\measuring@false\label@in@display{##1}\endgroup}%
2349     \def\math@cr@@@{\cr}%
2350     \let\shoveleft\@iden \let\shoveright\@iden
2351     \savecounters@
2352     \global\row@\z@
2353     \setbox\@ne\vbox{%
2354         \global\let\df@tag\@empty
2355         \halign{%
2356             \setboxz@h{\@lign$\m@th\displaystyle{##1}}%
```

```

2357         \iftagsleft@
2358             \ifnum\row@=\@ne
2359                 \global\totwidth@\wdz@
2360                 \global\lineht@\ht\z@
2361             \fi
2362         \else
2363             \global\totwidth@\wdz@
2364             \global\lineht@\dp\z@
2365         \fi
2366         \crrc
2367         #1%
2368         \crrc
2369     }%
2370 }%
2371 \ifx\df@tag\@empty\else\global\tag@true\fi
2372 \if@eqnsw\global\tag@true\fi
2373 \iftag@
2374     \setboxz@h{%
2375         \if@eqnsw
2376             \stepcounter{equation}%
2377             \tagform@\theequation
2378         \else
2379             \df@tag
2380         \fi
2381     }%
2382     \global\tagwidth@\wdz@
2383     \dimen@\totwidth@
2384     \advance\dimen@\tagwidth@
2385     \advance\dimen@\multlinetaggap
2386     \iftagsleft@\else
2387         \if@fleqn
2388             \advance\dimen@\@mathmargin
2389         \fi
2390     \fi
2391     \ifdim\dimen@>\displaywidth
2392         \global\shifftag@true
2393     \else
2394         \global\shifftag@false
2395     \fi
2396 \fi
2397 \restorecounters@
2398 \endgroup
2399 }

```

`\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.

```

2400 \iftagsleft@
2401     \def\shoveright#1{%
2402         #1%
2403         \hfilneg
2404         \hskip\multlinegap

```

```

2405     }
2406 \else
2407     \def\shoveright#1{%
2408         #1%
2409         \hfilneg
2410         \iftag@
2411             \ifshifftag@
2412                 \hskip\multlinegap
2413             \else
2414                 \hskip\tagwidth@
2415                 \hskip\multlinetaggap
2416             \fi
2417         \else
2418             \hskip\multlinegap
2419         \fi
2420     }
2421 \fi
2422
2423 \if@fleqn
2424     \def\shoveleft#1{#1}%
2425 \else
2426     \iftagsleft@
2427         \def\shoveleft#1{%
2428             \setboxz@h{$\m@th\displaystyle{#1}$}%
2429             \setbox\@ne\hbox{$\m@th\displaystyle#1$}%
2430             \hfilneg
2431             \iftag@
2432                 \ifshifftag@
2433                     \hskip\multlinegap
2434                 \else
2435                     \hskip\tagwidth@
2436                     \hskip\multlinetaggap
2437                 \fi
2438             \else
2439                 \hskip\multlinegap
2440             \fi
2441             \hskip.5\wd\@ne
2442             \hskip-.5\wdz@
2443             #1%
2444         }
2445     \else
2446         \def\shoveleft#1{%
2447             \setboxz@h{$\m@th\displaystyle{#1}$}%
2448             \setbox\@ne\hbox{$\m@th\displaystyle#1$}%
2449             \hfilneg
2450             \hskip\multlinegap
2451             \hskip.5\wd\@ne
2452             \hskip-.5\wdz@
2453             #1%
2454         }
2455     \fi
2456 \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.`

```
2457 \@saveprimitive\leqno\@leqno
2458 \@saveprimitive\eqno\@eqno
2459 \def\eqno{\@eqno\let\eqno\relax\let\leqno\relax}
2460 \def\leqno{\@leqno\let\leqno\relax\let\eqno\relax}
2461 %
2462 \let\veqno=\@eqno
2463 \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.

```
2464 \@ifundefined{SK@@label}{%
2465   \let\SK@@label\relax \let\SK@equationtrue\relax
2466 }{}
2467 \let\reset@equation\@empty
```

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

```
2468 \let\alt@tag\@empty
2469 \def\tag@in@display#1#{\relax\tag@in@display@a{#1}}
2470 \def\tag@in@display@a#1#2{%
2471   \iftag@
2472     \invalid@tag{Multiple \string\tag}\relax
2473   \else
2474     \global\tag@true \nonumber \reset@equation \st@rredtrue
2475     \if *string#1%
2476       \gdef\alt@tag{\def\SK@tagform@{#2\@gobble}}%
2477       \ifx\SK@@label\relax \let\tagform@\SK@tagform@ \fi
2478     }%
2479     \make@df@tag@{#2}%
2480   \else
2481     \make@df@tag@@{#2}%
2482   \fi
2483 \fi
2484 }
2485 \let\restore@hfuzz\@empty
2486 \def\mathdisplay#1{%
2487   \ifmmode \@badmath
2488   \else
2489     $$\def\@currenvir{#1}%
```

Allow use of `\displaybreak`.

```
2490 \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.

```
2491 \let\tag\tag@in@display \let\label\label@in@display \SK@equationtrue
2492 \global\let\df@label\@empty \global\let\df@tag\@empty
2493 \global\tag@false
2494 \let\mathdisplay@push\mathdisplay@@push
2495 \let\mathdisplay@pop\mathdisplay@@pop
2496 \if@fleqn
```

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

```
2497 \edef\restore@hfuzz{\hfuzz\the\hfuzz\relax}%
2498 \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.

```

2499     \setbox\z@\hbox to\displaywidth\bgroup
2500         \let\split@warning\relax \restore@hfuzz
2501         \everymath\@emptytoks \m@th $\displaystyle
2502     \fi
2503 \fi
2504 }

```

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

```

2505 \def\endmathdisplay#1{%
2506     \ifmmode \else \@badmath \fi
2507     \endmathdisplay@a
2508     $$%

```

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]

```

2509 \global\let\df@label\@empty \global\let\df@tag\@empty
2510 \global\tag@false \global\let\alt@tag\@empty
2511 \global\@eqnswfalse
2512 }

2513 \def\endmathdisplay@a{%
2514     \if@eqnsw \gdef\df@tag{\tagform@\theequation}\fi
2515     \if@fleqn \exp\endmathdisplay@fleqn
2516     \else \ifx\df@tag\@empty \else \veqno \alt@tag \df@tag \fi
2517     \ifx\df@label\@empty \else \exp\ltx@label\exp{\df@label}\fi
2518     \fi
2519     \ifnum\dspbrk@lvl>\m@ne
2520         \postdisplaypenalty -\@getpen\dspbrk@lvl
2521         \global\dspbrk@lvl\m@ne
2522     \fi
2523 }

```

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

```

2524 \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.

```

2525 \def\endmathdisplay@fleqn{%
2526     $\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).

```

2527 \ifnum\badness<\inf@bad \let\too@wide\@ne \else \let\too@wide\z@ \fi
2528 \ifx\@empty\df@tag
2529 \else
2530 \setbox4\hbox{\df@tag
2531 \ifx\df@label\@empty \else \@xp\ltx@label\@xp{\df@label}}\fi
2532 }%
2533 \fi
2534 \csname emdf@%
2535 \ifx\df@tag\@empty U\else \iftagsleft@ L\else R\fi\fi
2536 \endcsname
2537 }

```

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.

```

2538 \def\emdf@U{%
2539 \restore@hfuzz
2540 \ifodd\too@wide % not too wide: just need to swap the glue around
2541 \hbox to\displaywidth{\hskip\@mathmargin\unhbox\z@\unskip}}%
2542 \else % M+B > displaywidth
2543 \emdf@Ua
2544 \fi
2545 }

```

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.

```

2546 \def\emdf@Ua{%
2547 \hbox to\columnwidth{%
2548 \ifdim\displayindent>\z@
2549 \hskip\displayindent minus\displayindent
2550 \fi
2551 \hskip\@mathmargin \unhbox\z@ \unskip
2552 }%
2553 \displayindent\z@ \displaywidth\columnwidth
2554 }

```

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.

```

2555 \def\emdf@R{%
2556 \setbox\tw@\hbox to\displaywidth{%
2557 \hskip\@mathmargin \unhcopy\z@\unskip\hfil\hskip\mintagsep\copy4
2558 }%
2559 \restore@hfuzz
2560 \ifnum\badness<\inf@bad \box\tw@ \else \emdf@Ra \fi
2561 }

```

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.

```

2562 \def\emdf@Ra{%
2563   \skip@\displayindent minus\displayindent
2564   \displayindent\z@ \displaywidth\columnwidth
2565   \spread@equation \everycr{}\tabskip\z@skip
2566   \halign{\hbox to\displaywidth{##}\cr
2567     \relax
2568     \ifdim\skip@>\z@ \hskip\skip@ \fi
2569     \hskip\@mathmargin\unhbox\z@\unskip\hfil\cr
2570     \noalign{\raise@tag}%
2571     \hfil\box4 \cr}%
2572 }

```

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.

```
2573 \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 $x \text{ minus } x$ where $x = M - (N + S)$. I.e., the distribution of spaces across the line is $N, S, x \text{ minus } x, B, h \text{ fil}$.

```

2574   \@tempdima\@mathmargin
2575   \advance\@tempdima-\wd4 \advance\@tempdima-\mintagsep
2576   \skip@\@tempdima minus\@tempdima
2577   \setbox\tw\hbox to\displaywidth{%
2578     \copy4\hskip\mintagsep
2579     \ifdim\skip@>\z@ \hskip\skip@\fi
2580     \unhcopy\z@\unskip
2581   }%
2582   \restore\hfuzz
2583   \ifnum\badness<\inf@bad \box\tw@ \else \emdf@La \fi
2584 }

```

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.

```

2585 \def\emdf@La{%
2586   \spread@equation \everycr{}\tabskip\z@skip
2587   \halign{\hbox to\displaywidth{##}\cr
2588     \box4 \hfil \cr
2589     \noalign{\raise@tag}%
2590     \hskip\@mathmargin\unhbox\z@\unskip\hfil\cr}%
2591 }

```

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.

```

2592 \newtoks\mathdisplay@stack
2593 \let\mathdisplay@push\@empty
2594 \def\mathdisplay@push{%
2595   \begingroup
2596   \toks@\@xp{\df@label}\@temptokena\@xp{\df@tag}%
2597   \toks8\@xp{\alt@tag}%
2598   \edef\@tempa{%
2599     \global\if@eqnsw\@nx\@eqnswtrue\else\@nx\@eqnswfalse\fi
2600     \global\iftag@\@nx\tag@false\else\@nx\tag@true\fi
2601     \gdef\@nx\df@label{\the\toks@}\gdef\@nx\df@tag{\the\@temptokena}%

```

```

2602   \gdef\@nx\alt@tag{\the\toks8}%
2603   \global\mathdisplay@stack{\the\mathdisplay@stack}%
2604   }%
2605   \global\mathdisplay@stack\@xp{\@tempa}
2606   \endgroup
2607 }

2608 \let\mathdisplay@pop\@empty
2609 \def\mathdisplay@@pop{\the\mathdisplay@stack}

2610 \renewenvironment{equation}{%
2611   \incr@eqnum
2612   \mathdisplay@push
2613   \st@rredfalse \global\@eqnswtrue
2614   \mathdisplay{equation}%
2615 }{%
2616   \endmathdisplay{equation}%
2617   \mathdisplay@pop
2618   \ignorespacesafterend
2619 }

2620 \newenvironment{equation*}{%
2621   \mathdisplay@push
2622   \st@rredtrue \global\@eqnswfalse
2623   \mathdisplay{equation*}%
2624 }{%
2625   \endmathdisplay{equation*}%
2626   \mathdisplay@pop
2627   \ignorespacesafterend
2628 }

```

Note: L^AT_EX defines the `displaymath` environment in terms of `\[` and `\]`.

```

2629 \DeclareRobustCommand{\[}{\begin{equation*}}
2630 \DeclareRobustCommand{\]}{\end{equation*}}

```

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

```
2631 \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 `\'` 10, 16 `\C` 10, 16, 493
`\"` 10, 16, 285

- \) 10, 16
- * 10, 16
- \+ 10, 16
- \, 10, 16, 38, 102, 103,
340, 341, 342,
343, 409, 411,
415, 429, 430,
431, 673, 765,
771, 1152, 1216
- \- 10, 16, 697, 700
- \. 10, 16
- .toc 6
- \: 106, 107
- \; 109, 110,
263, 264, 265, 448
- \< 11, 17
- \= 11, 17, 698, 701
- \> 11, 17
- \@@above 147
- \@@abovewithdelims 150
- \@@atop 146
- \@@atopwithdelims 149
- \@@eqno 2458, 2459, 2462
- \@@italiccorr 923
- \@@leqno 2457, 2460, 2463
- \@@cover 6, 145, 166
- \@@overline .. 260, 261
- \@@overwithdelims 148
- \@@sqrt 8, 240
- \@@undefined . 123, 138
- \@addtoreset 850
- \@amsmath@err
. 90, 214, 215,
566, 664, 789,
874, 879, 897,
917, 941, 1183,
1444, 1544, 2061
- \@arrayboxrestore 947
- \@arrayparboxrestore
..... 946, 989
- \@backslashchar .. 153
- \@badmath .. 2487, 2506
- \@car 94
- \@cdots ... 11, 376,
377, 378, 421, 456
- \@centering
. 2, 40, 1265, 2265
- \@checkend 1134
- \@currentlabel ...
.... 30, 934, 936
- \@currenvir
... 783, 1120,
1123, 1124,
1149, 1229,
1432, 2228, 2489
- \@display@init ...
35, 66, 1050, 2250
- \@displaytrue 452
- \@eha 214,
215, 874, 897, 942
- \@ehd 665
- \@elt 1084
- \@empty 19
- \@emptytoks
95, 267, 1121, 2501
- \@envbody 36,
37, 1116, 1117,
1119, 1120, 1121
- \@eqnum .. 30, 925, 928
- \@eqnswfalse .. 983,
2511, 2599, 2622
- \@eqnswtrue
.. 1233, 1311,
1512, 1531,
1560, 2074,
2235, 2599, 2613
- \@eqpen 27, 35, 50, 68,
68, 886, 887, 1061
- \@fleqnfalse ... 55, 57
- \@fleqntrue . 43, 52, 53
- \@genfrac .. 7, 174, 178
- \@getpen . 27, 889, 2520
- \@gobblethree 572, 578
- \@gtempa 1086,
1091, 1098, 1105
- \@highpenalty 27
- \@iden 2350
- \@ifnotempty . 745, 747
- \@ifpackagewith ... 55
- \@ignoretrue 834
- \@ldots 10, 332, 340,
409, 411, 425,
429, 430, 431, 810
- \@let@token 65
- \@lign .. 1528, 1551,
2037, 2041, 2356
- \@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, 1296,
1319, 1323,
1327, 1622,
1642, 1654,
1671, 1694,
1715, 1730,
2257, 2260,
2275, 2276,
2297, 2388,
2526, 2541,
2551, 2557,
2569, 2574, 2590
- \@mathmeasure .. 95,
513, 682, 685, 686
- \@mathstyle .. 174, 179
- \@medpenalty 27
- \@nocnterr 849
- \@nocounterr 848
- \@saveprimitive ..
145, 146, 147,
148, 149, 150,
260, 2457, 2458
- \@spread@equation
..... 991, 1049
- \@sptoken 223, 229,
294, 299, 315, 323
- \@temp 4, 7, 9, 14
- \@tempa 60, 60, 60
- \@tempcnta
50, 54, 59, 60, 60
- \@tempcntb . 50, 54,
54, 58, 58, 60, 60
- \@tempd 544
- \@tempdima 53,
55, 58, 58, 60,
60, 1715, 1716,
1738, 1778,
1779, 1780,
1784, 1791,
1795, 1843,
1844, 1851,
1857, 1862,
1870, 1924,
1933, 1934,
1935, 1942,
1953, 1964,
1986, 1988,
2574, 2575, 2576
- \@tempdimb 54,
58, 58, 59,
60, 60, 1762,
1827, 1893,
1896, 1899,
1900, 1903,
1925, 1934,
1950, 1976,
1990, 1991, 1999
- \@tempdimc 54,
1763, 1779,
1828, 1843,
1893, 1895, 1898
- \@testopt 39, 1197, 1206
- \@totalleftmargin
..... 28, 28, 905
- \[3, 11, 17, 70, 75, 2629
- \ \ . . 23, 23, 26, 26,
27, 28, 28, 28,
30, 32, 38, 280,
308, 321, 894, 900
- \ \ * 23
- \} 383
- \] 3, 11, 17, 70, 75, 2630
- \^ 11, 18

- `_` 11, 18
- `\'` 5
- `\sqcup` 345
- A**
- `\above` 147, 162
- `\abovedisplayskip` 908
- `\abovewithdelims` .
..... 150, 165
- `\acc@check`
. 19, 562, 564, 639
- `\acc@error` 563, 564, 565
- `\accentclass@` .. 15,
520, 521, 584, 610
- accents package 18
- `\Acute` 657
- `\acute` 548, 657
- `\add@amp` 1006
- `\add@amps` 63, 1006,
1524, 1606, 2145
- `\addto@envbody` ...
. 1117, 1135, 1137
- `\addto@hook` 499, 506, 516
- `\addtocounter` 980
- `\aftergroup` . 2162, 2196
- align environment .
... 31, 31, 32,
34, 36, 39, 44,
44, 44, 44, 44,
45, 48, 49, 51,
52, 52, 60, 62,
63, 63, 63, 68, 1450
- align* environment
..... 45, 1450
- `\align@` 44,
1425, 1437, 1505
- `\align@lengths` . 50, 50
- `\align@preamble` ..
. 1517, 1602, 2034
- `\align@recover` ...
..... 1429, 1442
- align[at] environ-
ment 75
- alignat environment
..... 44,
44, 47, 47, 48,
49, 50, 50, 50, 1450
- alignat* environment
..... 1450
- aligned environment
..... 38,
38, 39, 39, 39,
39, 45, 65, 65, 1196
- `\aligned@a` . 1198, 1204
- alignedat environ-
ment ... 38,
38, 38, 39,
39, 39, 39, 1205
- `\alignedat@a` 1207, 1211
- `\alignsafe@testopt`
. 1190, 1197, 1206
- `\alignsep@` 49,
50, 50, 50, 50,
50, 50, 51, 53,
53, 53, 54, 54,
54, 55, 55, 55,
55, 58, 59, 59,
59, 59, 1018,
1143, 1158,
1161, 1176,
1526, 1591,
1637, 1655,
1656, 1661,
1669, 1671,
1673, 1675,
1676, 1681,
1691, 1731,
1732, 1733,
1734, 1735,
1789, 1801,
1803, 1858,
1868, 1871,
1881, 1882,
1947, 1962,
1965, 1968,
1969, 1977, 2045
- `\allowbreak` .. 669, 672
- `\allowdisplaybreaks`
..... 27, 858
- `\alph` 841
- `\alt@tag` 2468,
2476, 2510,
2516, 2597, 2602
- `\AmS` 4, 91
- amsbsy package 4, 4
- `\AmSfont` 4, 91, 93
- 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, 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` 265
- `\andhelp@` 34,
1014, 1184, 1545
- `\arabic` 847
- array environment 23,
24, 24, 24, 25, 26
- `\array` 24, 780, 830
- `\arraycolsep` 24, 776, 778
- `\arraystretch` 829
- `\arrowfill@`
705, 711, 712,
713, 714, 715, 716
- `\AtBeginDocument` .
..... 46, 507, 699
- `\AtEndOfPackage` 8
- `\atop` 6, 23, 23, 146, 161
- `\atopwithdelims` 149, 164
- B**
- `\badness` 72,
2527, 2560, 2583
- `\Bar` 662
- `\bar` 553, 662
- `\bBigg` 14
- `\bBigg@` 508,
509, 510, 511, 512
- `\begin` ... 28, 37, 37, 37
- `\begin/endgroup` ... 68
- `\begin@stack`
. 1121, 1131, 1132
- `\begingroup` 28
- `\belowdisplayskip` 902
- `\bgroup` 235,
484, 753, 759,
765, 768, 1153,
1167, 1217,
1220, 1260,
1425, 1516,
2097, 2105,
2108, 2262, 2499
- `\Big` 15, 508
- `\big` 15, 15, 508
- `\big@size` ... 514, 516
- `\bigcap` 196, 197
- `\bigcap@` 196, 197
- `\bigcup` 198, 199
- `\bigcup@` 198, 199
- `\Bigg` 508
- `\bigg` 508
- `\Biggr` 394
- `\biggr` 392
- `\bigl` 15, 15
- `\bigodot` 208, 209
- `\bigodot@` ... 208, 209
- `\bigoplus` ... 206, 207

- `\bigoplus@` .. 206, 207
`\bigotimes` .. 204, 205
`\bigotimes@` .. 204, 205
`\Bigr` 393
`\bigr` 15, 391
`\bigsqcup` ... 210, 211
`\bigsqcup@` .. 210, 211
`\biguplus` ... 194, 195
`\biguplus@` .. 194, 195
`\bigvee` 190, 191
`\bigvee@` 190, 191
`\bigwedge` ... 192, 193
`\bigwedge@` .. 192, 193
`\binom` 6, 6, 169
`\binrel@` .. 21, 677, 679
`\binrel@@` . 21, 678, 680
`\black` 52
`\black@` 1070,
1241, 1478, 2100
bm package 4
`\bmod` 20, 666
`\bold@false` 375
`\bold@true` 374
`\boldsymbol` . 4, 348, 375
`\boldsymboldots@` .
..... 11, 348, 373
`\boxed` 9, 262
`\boxz@` 64,
246, 1366, 1372,
1387, 1391,
1403, 2010,
2015, 2027,
2030, 2058,
2133, 2152, 2155
`\Breve` 661
`\breve` 552, 661
`\buildrel` 13
- C**
- `\c@MatrixCols` 24
`\c@MaxMatrixCols` .
..... 780, 781
`\cal` 15
`\calc@shift@align`
..... 52,
54, 1689, 1711,
1745, 1810, 1907
`\calc@shift@gather`
33, 42, 1263, 1315
cases environment 25, 821
`\cases` 25, 822
`\cdotp` 376, 457, 458, 459
`\cdots` 11,
12, 421, 422, 423
`\cfrac` 674
`\Check` 655
`\check` 546, 655
`\check@mathfonts` .. 14
`\checkat@false` .. 1421
`\checkat@true` ... 1415
`\cl@ckpt` 1087
`\classnum@` 288, 289, 290
`\cleaders` 708
cmex 2
cmex10 4
cmex10 option 2, 3
`\cmex@opt` ... 3, 33,
34, 36, 70, 77, 84
`\collect@body` ...
..... 37, 37,
37, 37, 1123, 1130
`\collect@body` .. 37,
37, 1118, 1236,
1440, 2076,
2078, 2200, 2239
`\colon` 9, 272
`\column@` 33,
35, 39, 39, 54,
54, 60, 1001,
1003, 1009,
1067, 1111,
1142, 1166,
1180, 1187,
1280, 1540,
1548, 1570,
1598, 1605,
1761, 1766,
1770, 1771,
1772, 1774,
1775, 1826,
1831, 1835,
1836, 1837,
1839, 1840,
1894, 1895,
1899, 1903,
1923, 1928,
1989, 1990,
1993, 1999, 2138
`\column@plus`
..... 33, 1002,
1168, 1173, 2048
`\columnwidth` ... 28, 73
`\coprod` 7, 186, 188, 189
`\coprod@` 188, 189
`\copy` 245, 490,
501, 505, 685,
741, 2557, 2578
`\count@` .. 58, 59, 60, 60
`\cr` 23, 28,
895, 1145, 1188,
1313, 1533,
1561, 1606,
2349, 2566,
2569, 2571,
2587, 2588, 2590
`\crr` ... 63, 68, 68,
717, 718, 726,
727, 760, 762,
769, 771, 1177,
1200, 1222,
1269, 1289,
1517, 1602,
2080, 2114,
2143, 2157,
2273, 2366, 2368
`\ctagsplit@false` .. 31
`\ctagsplit@true` ... 30
- D**
- `\dag` 9
`\dbinom` 6, 170
`\ddddot` 527
`\dddot` 524
`\Ddot` 660
`\ddot` 551, 660
`\DeclareFontEncoding`
..... 522, 523
`\DeclareFontShape` . 71
`\DeclareMathAccent`
..... 16, 18, 556
`\DeclareMathDelimiter`
125, 127, 139, 141
`\DeclareMathOperator`
..... 4
`\DeclareMathSymbol`
..... 248,
249, 250, 251,
252, 253, 254,
255, 256, 257, 258
`\DeclareOption` ...
19, 20, 21, 22,
23, 24, 28, 29,
30, 31, 32, 42, 52
`\DeclareRobustCommand`
..... 5, 5, 100,
151, 166, 169,
172, 261, 333,
336, 421, 424,
428, 2629, 2630
`\default@tag` 23, 29,
754, 766, 918,
990, 1164, 2107
`\delayed@` ... 374, 408
`\delimiter` ... 114,
119, 121, 134, 136
`\df@label` 30,
31, 937, 955,
957, 958, 2492,
2509, 2517,
2531, 2596, 2601
`\df@tag` 30,
30, 31, 31, 931,
951, 1556, 2089,
2091, 2354,
2371, 2379,
2492, 2509,
2514, 2516,

- 2528, 2530, 35, 35, 39, 66, 316, 319, 324,
 2535, 2596, 2601, 1050, 1254, 1507, 325, 340, 411,
 \dfrac 6, 6, 167 \displaybreak 26, 27, 415, 421, 425,
 \dft@tag 29, 918 50, 68, 68, 71, 864 429, 965, 1229,
 \dimen@ . 51, 59, 59, \displayindent 1425, 1429,
 59, 59, 59, 242, 73, 73, 1432, 1437, 2228
 244, 245, 637, 74, 2548, 2549, docstrip 75
 638, 651, 652, 2553, 2563, 2564 \Dot 659
 687, 688, 1073, \displaylimits . 7, \dot 550, 659
 1078, 1316, 7, 19, 21, 477, 478 \doteq 13, 449
 1321, 1323, displaymath environ- \dots 10, 11, 336
 1327, 1328, ment . 70, 71, 75 \DOTS@ 309,
 1329, 1338, \displaystyle . 181, 315, 319, 362, 401
 1339, 1340, 238, 262, 674, \DOTS@@ 310, 314
 1341, 1376, 682, 685, 686, \DOTSB 7,
 1377, 1378, 737, 1171, 189, 191, 193,
 1634, 1635, 1174, 1221, 195, 197, 199,
 1645, 1655, 1262, 1283, 201, 203, 205,
 1659, 1669, 2037, 2041, 207, 209, 211,
 1702, 1703, 2111, 2112, 263, 264, 265,
 1781, 1782, 2270, 2356, 278, 435, 437,
 1783, 1784, 2428, 2429, 439, 441, 442,
 1785, 1789, 2447, 2448, 2501 443, 444, 445,
 1790, 1791, \displaywidth . . 35, 446, 447, 448, 450
 1792, 1793, 42, 50, 52, 55, \dotsb 422
 1794, 1795, 55, 58, 58, 73, \dotsb@ . . 290, 291,
 1797, 1799, 73, 926, 1072, 350, 352, 361,
 1800, 1803, 1260, 1299, 364, 368, 370, 378
 1845, 1846, 1302, 1329, \dotsc 349, 428
 1847, 1849, 1335, 1341, \DOTSCASE@ 307, 317,
 1851, 1852, 1352, 1375, 318, 319, 364, 403
 1856, 1857, 1377, 1634, \DOTSI 277, 453, 454,
 1858, 1859, 1679, 1700, 461, 462, 463, 464
 1860, 1861, 1704, 1707, \dotssi 365, 377
 1862, 1864, 1731, 1785, \dotism 423
 1866, 1867, 1793, 1794, \dotso 424
 1874, 1875, 1852, 1860, \dotso@ 347, 406
 1878, 1881, 1861, 1869, \dotsspace@
 1882, 1936, 1943, 1952, 813, 816, 817, 819
 1937, 1938, 1963, 2258, \DOTSX 279
 1940, 1942, 2391, 2499, \dp 242, 497, 518, 641,
 1943, 1947, 2541, 2553, 683, 691, 1076,
 1948, 1949, 2556, 2564, 1385, 1400,
 1950, 1951, 2566, 2577, 2587 2025, 2054,
 1952, 1953, \displaywidth@ 2055, 2329, 2364
 1955, 1958, . 1705, 1707, 2017 \dspbrk@ 864
 1959, 1961, \divide 289, \dspbrk@context 864,
 1963, 1964, 1349, 1646, 1163, 1218,
 1965, 1966, 1650, 1656, 1253, 1506,
 1968, 1969, 1660, 1665, 2106, 2251, 2490
 1973, 1975, 1673, 1682, \dspbrk@lvl . 26, 27,
 1976, 1977, 1733, 1771, 35, 855, 870,
 1978, 1979, 1799, 1836, 888, 889, 1061,
 2297, 2298, 1866, 1872, 1067, 2250,
 2299, 2301, 1958, 1966, 1995 2519, 2520, 2521
 2383, 2384, \DN@ 219, 223, \dt@pfalse 1057
 2385, 2388, 2391 224, 230, 231, \dt@ptrue 1052
 \disply . . 35, 66, 1050 268, 295, 296,
 \disply@ 300, 301, 309,

- E**
- `\edef` 26
 - `\egroup` 72, 239, 486, 762, 771, 1200, 1202, 1241, 1479, 1482, 2081, 2082, 2100, 2101, 2317, 2342, 2526
 - `\emdf@L` 2573
 - `\emdf@La` . . . 2583, 2585
 - `\emdf@R` 2555
 - `\emdf@Ra` . . . 2560, 2562
 - `\emdf@U` 2538
 - `\emdf@Ua` . . . 2543, 2546
 - `\end` 37, 37, 37, 37
 - `\endalign` . . . 1453, 1458, 1463, 1468, 1473, 1493, 1498, 1503
 - `\endaligned` . 1209, 1224
 - `\endarray` . 24, 776, 824
 - `\endgather` 1248
 - `\endinput` 75
 - `\endmathdisplay` . . . 2505, 2616, 2625
 - `\endmathdisplay@a` . 2099, 2507, 2513
 - `\endmathdisplay@fleqn` 2515, 2525
 - `\endmatrix` . . . 800, 802, 806, 808, 809
 - `\endmultline` 2247
 - `\endmultline@math` . 2270, 2312, 2322
 - `\endsplit` 2076
 - `\endsubarray` 764
 - `\env@cases` . . 822, 826
 - `\env@matrix` 774, 778, 798, 802, 804, 808, 809
 - environments:
 - `align` 31, 31, 32, 34, 36, 39, 44, 44, 44, 44, 44, 45, 48, 49, 51, 52, 52, 60, 62, 63, 63, 63, 68, 1450
 - `align*` . . . 45, 1450
 - `align[at]` 75
 - `alignat` 44, 44, 47, 47, 48, 49, 50, 50, 50, 1450
 - `alignat*` 1450
 - `aligned` 38, 38, 39, 39, 39, 39, 45, 65, 65, 1196
 - `alignedat` 1686, 1687, 1697, 1729, 1730, 1778, 1856, 1869, 1870, 1871, 1872, 1874, 1876, 1878, 1949, 1978, 1979
 - `alignedat` 38, 38, 38, 39, 39, 39, 39, 1205
 - `array` 23, 24, 24, 24, 25, 26
 - `cases` 25, 821
 - `displaymath` 70, 71, 75
 - `eqnarray` . . . 26, 30
 - `equation` . . . 70, 70
 - `equation*` 70
 - `flalign` . 44, 48, 49, 51, 52, 59, 1450
 - `flalign*` 1450
 - `gather` 32, 33, 34, 36, 38, 40, 62, 63, 1238
 - `gather*` 1238
 - `gathered` . 40, 1212
 - `matrix` . . 24, 24, 773
 - `multiline` 33, 35, 65, 66, 69, 69, 2241
 - `multline*` 2241
 - `pmatrix` 24
 - `Sb` 23
 - `smallmatrix` 23, 765
 - `Sp` 23
 - `split` 42, 61, 61, 62, 63, 63, 63, 63, 2069
 - `subarray` 23, 23, 23, 23, 752
 - `xalignat` 44, 44, 51, 1450
 - `xalignat*` 1450
 - `xxalignat` . . . 44, 44, 48, 49, 51, 1450
 - `eqnarray` environment 26, 30
 - `\eqnarray` 3
 - `\eqno` 72, 2458, 2459, 2460
 - `\eqnshift@` . 34, 49, 50, 52, 53, 53, 54, 55, 58, 59, 59, 59, 59, 59, 1017, 1319, 1335, 1336, 1344, 1345, 1349, 1351, 1352, 1353, 1355, 1356, 1402, 1515, 1590, 1642, 1645, 1646, 1654, 1659, 1660, 1661, 1668, 1677, 1679, 1680, 1681, 1682,
 - `\eqref` 26, 854
 - equation environment 70, 70
 - `\equation` . . . 3, 30, 44
 - `equation*` environ-
ment 70
 - `\every@math@size` 14, 14, 499, 506, 516
 - `\everycr` 35, 66, 1053, 1279, 1597, 2565, 2586
 - `\everydisplay` 452
 - `\everymath` 2501
 - `\ex@` . 525, 528, 727, 767
 - `\ExecuteOptions` . . . 53
 - `\expandafter` 37
 - `\ext@arrow` 734, 735, 750, 751
 - `\extra@` 396, 409, 411, 418, 431
 - `\extrap@` . 414, 421, 425
- F**
- `\f@series` 4, 4, 94
 - `\fbox` 9, 262
 - `\fi` 28, 28
 - `\field@lengths` 1563, 1593
 - `\fieldlengths@` 54, 1563, 1764, 1829, 1926
 - `\finsm@sh` 692
 - `flalign` environment 44, 48, 49, 51, 52, 59, 1450
 - `flalign*` environment 1450
 - `fleqn` option 2, 3, 42, 48, 48, 48, 50, 50, 51, 51, 51, 51, 65, 67, 69, 75
 - `fleqn.clo` 3, 3, 3
 - `\FN@` . . 222, 223, 224, 228, 230, 231, 235, 267, 270, 296, 301, 316, 325, 345, 346, 411, 421, 427, 433
 - `\fontdimen` 755, 756, 757, 1020
 - `\fontencoding` 79

- `\fontfamily` 79
`\frac` 6, 6, 154, 166, 674
`\frozen@everymath`
 95, 579, 599
- G**
- gather environment
 32, 33, 34, 36,
 38, 40, 62, 63, 1238
gather* environment 1238
`\gather@` 1234, 1250, 2098
`\gather@split` 2078, 2086
gathered environment
 40, 1212
`\gdisplaywidth@` . .
 1268,
 1300, 1302, 1362
`\genfrac` 6, 6,
 154, 167, 168,
 169, 170, 171, 172
`\getdsp@pen` . . 859, 861
`\getmathch@` . . 289, 358
`\glb@settings` 14
`\global` 9, 25
`\gmeasure@` . . 1256, 1272
`\gobble@tag`
 . . . 916, 917, 2256
`\Grave` 658
`\grave` 549, 658
`\gtest@false` 9, 276,
 282, 286, 295,
 300, 303, 309,
 324, 331, 394, 403
`\gtest@true`
 . . 9, 275, 283,
 287, 294, 299,
 304, 315, 323,
 327, 379, 398, 403
- H**
- `\halign` 13, 23,
 48, 48, 50, 50,
 50, 58, 1260,
 1282, 1516,
 1602, 2262,
 2355, 2566, 2587
`\Hat` 20, 654
`\hat` 17, 20, 545, 654
`\hbox` 50, 92,
 95, 236, 243,
 489, 492, 503,
 526, 529, 576,
 635, 636, 639,
 640, 688, 708,
 737, 739, 740,
 743, 817, 922,
 925, 1077, 1627,
 2123, 2132,
 2136, 2158,
- 2171, 2180, 2259, 2264,
 2185, 2263, 2274, 2387,
 2287, 2296, 2423, 2496, 2515
 2331, 2429, `\ifbold@` 405, 407
 2448, 2499, `\ifcase` 47
 2530, 2541, `\ifcheckat@` . . 1406, 1541
 2547, 2556, `\ifctagsplit@`
 2566, 2577, 2587 26, 2130, 2151,
 2178, 2190, 2210
`\hdots` 810
`\hdots@for` 812, 814, 815
`\hdotsfor` 811
`\hexnumber@` 532
`\hfil` 66, 66, 67, 718,
 726, 759, 760,
 768, 769, 1169,
 1175, 1221,
 1404, 2035,
 2044, 2109,
 2113, 2267,
 2271, 2526,
 2557, 2569,
 2571, 2588, 2590
`\hfill` . . . 675, 708,
 818, 1077, 2113
`\hfilneg` 66,
 2278, 2314,
 2340, 2403,
 2409, 2430, 2449
`\hfuzz` 2497, 2498
`\hookleftarrow` . . . 447
`\hookrightarrow` . . 446
`\hskip` 66
`\hspace` 5
`\ht` 242, 497,
 517, 641, 683,
 691, 1075, 1398,
 2050, 2051, 2360
- I**
- `\ialign` 717,
 726, 759, 768,
 1167, 1220, 2108
`\idotsint` 464
`\if` 4
`\if@display`
 . . 13, 451, 670,
 672, 1428, 2070
`\if@eqnsw` 950,
 979, 1522, 1539,
 1552, 2372,
 2375, 2514, 2599
`\if@fleqn` 37,
 1295, 1318,
 1622, 1640,
 1653, 1670,
 1678, 1692,
 1711, 1745,
 1810, 1907,
 2205, 2257,
- 2259, 2264,
 2274, 2387,
 2423, 2496, 2515
`\ifbold@` 405, 407
`\ifcase` 47
`\ifcheckat@` . . 1406, 1541
`\ifctagsplit@`
 26, 2130, 2151,
 2178, 2190, 2210
`\ifdt@p` 1056
`\iff` 448
`\ifgtest@` 9,
 274, 275, 276,
 352, 356, 358,
 360, 361, 363,
 367, 368, 370,
 397, 400, 402,
 409, 411, 419, 431
`\ifhbox` 1630
`\ifinalign@` 63,
 64, 987, 2117, 2166
`\ifinany@` 32
`\ifingather@` . . 987,
 1423, 1480, 1507
`\ifinner` 13, 2071
`\ifmeasuring@` . . 31,
 33, 867, 953,
 998, 2038, 2042
`\ifmmode` 5
`\ifshifftag@` . . . 33,
 999, 1363,
 1380, 2284,
 2323, 2411, 2432
`\ifst@rred` 997, 1233,
 1308, 1311,
 1512, 1521,
 1531, 1538,
 1560, 2074, 2232
`\iftag@` . . . 31, 951,
 964, 996, 1527,
 1556, 2088,
 2282, 2321,
 2373, 2410,
 2431, 2471, 2600
`\iftagsleft@` . . . 27,
 924, 974, 1268,
 1298, 1322,
 1350, 1361,
 1397, 1699,
 1711, 1745,
 1810, 1890,
 1907, 2005,
 2049, 2083,
 2212, 2244,
 2281, 2357,
 2386, 2400,
 2426, 2463, 2535
`\ifxxat@` . . . 1406, 1509

- `\ignorespacesafterend` 25, 834, 845, 1242, 1488, 2245, 2618, 2627
`\iiiint` 463
`\iiint` 462
`\iint` 461
`\ilimits@` 19, 20, 453, 454, 471, 478, 486
`\impliedby` 264
`\implies` 263
`\inalign@false` ... 990
`\inalign@true` ... 1506
`\incr@eqnum` 950, 980, 982, 986, 994, 2611
`\inf@bad` 5, 98, 2527, 2560, 2583
`\ingather@false` .. 990
`\ingather@true` .. 1251
`\insplit@` 1251, 1436, 2104
`\int` 453
`\intdots@` ... 456, 467
`\interdisplaylinepenalty` ... 26, 26, 35, 50, 857, 859, 888
`\intertext` 28, 28, 28, 897
`\intertext@` 28, 897, 1254, 1506
`\intkern@` 455, 467, 468, 469
`\intop` 453, 466, 468, 469, 470
`\ints@a` 473, 475
`\ints@b` 476, 477, 478, 482
`\ints@c` 466, 480
`\Invalid@` 214, 215, 874, 897
`\invalid@tag` 29, 31, 917, 918, 965, 2472
- J**
- `\joinrel` 435, 437, 439, 441, 442, 443, 446, 447
`\jot` 1048
- K**
- `\kern` 92, 101, 450, 505, 525, 528, 643, 652, 676, 678, 680, 688, 727, 1143, 1362, 1378, 1402, 1526, 2006, 2017, 2019
`\keybin@` 327, 351
- L**
- `\label` 30, 30, 30, 30, 30, 31, 31, 31, 31, 31, 31, 66, 66, 68
`\label@in@display` 30, 71, 937, 1252, 1510, 2348, 2491
`\lastbox` . 50, 1619, 1628, 2120, 2126, 2128, 2169, 2174, 2176
`\lbrace` 804, 828
`\ldots` 10, 333
`\left` . 514, 797, 802, 804, 808, 809, 828
`\Leftarrow` 441, 443, 714, 716
`\leftarrow` ... 439, 442, 447, 711, 713
`\Leftarrowfill@` .. 714
`\leftarrowfill@` .. 711, 722, 731, 751
`\leftmargini` 3, 48
`\Leftrightarrowfill@` 716
`\leftrightharpoonfill@` 713, 724, 733
`\leftroot` 214, 220, 225, 227, 228
`\leftroot@` 217, 227, 228, 235, 245, 246
`\lendmultline@` ... 66, 2244, 2313
`\lendsplit@` . 2083, 2164
`\legno` 72, 2457, 2459, 2460
`\let` 11, 37, 37
`\Let@` .. 23, 28, 754, 766, 894, 1163, 1218, 1254, 1506, 2106, 2249
`\lhook` 446
`\limits` 13, 23, 450, 476, 525, 528, 678, 680, 744
`\lineht@` ... 34, 61, 63, 1024, 1367, 1385, 1398, 1400, 1532, 2011, 2025, 2050, 2051, 2054, 2055, 2144, 2291, 2329, 2360, 2364
`\lineskip` 24
`\lineskiplimit` ... 24, 758, 767, 1058
- `\linewidth` 28, 28, 904, 905
`\llap` ... 1381, 1391, 2021, 2030, 2325
`\Longleftarrow` 264, 440
`\longleftarrow` ... 438
`\Longleftrightarrow` 443, 448
`\longleftrightarrow` 442
`\longmapsto` 445
`\Longrightarrow` 263, 436
`\longrightarrow` 434, 445
`\loop` ... 1626, 1717, 1749, 1814, 1911
`\lower` 92, 504
`\lq` 67
`\ltx@label` 30, 937, 957, 2517, 2531
`\lVert` 134, 138, 139, 809
`\lvert` 119, 123, 125, 808
- M**
- `\m@ne` . 615, 617, 632, 980, 1061, 1067, 1155, 1204, 1414, 1476, 1491, 1496, 1501, 1652, 1666, 1724, 1756, 1770, 1821, 1835, 1918, 1994, 2250, 2519, 2521
`\m@th` .. 29, 96, 237, 241, 243, 262, 340, 706, 718, 726, 738, 760, 768, 769, 816, 817, 922, 1171, 1174, 1221, 1262, 1283, 2037, 2041, 2111, 2112, 2270, 2356, 2428, 2429, 2447, 2448, 2501
`\macc@a` 611, 628
`\macc@adjust` 645, 646, 650
`\macc@code` 610, 643, 645
`\macc@depth` 571, 575, 582, 594, 632, 633
`\macc@group` 595, 599, 607, 608
`\macc@kerna` 638, 643, 651
`\macc@kernb` .. 642, 651
`\macc@nested` . 586, 596
`\macc@nested@a` ... 601, 602, 606

- \backslash maccnucleus 583, 589, 634
 \backslash maccpalette 19, 605, 611, 631
 \backslash macc@set@skewchar 19, 598, 600, 608, 613
 \backslash macc@skewchar 624, 635, 636
 \backslash macc@style 19, 630, 631
 \backslash macctest 577, 594
 \backslash macctmp 575, 581, 594
 \backslash macro@ 303, 359, 400
 \backslash macro@@ 305, 306
 \backslash make@df@tag 30, 30, 31, 931, 969
 \backslash make@df@tag@@ 931, 2479
 \backslash make@df@tag@@@ 931, 2481
 \backslash make@display@tag 31, 949, 1310, 1528, 2289, 2296, 2331, 2335
 \backslash makesm@sh 694
 \backslash maketag@@ 29, 30, 921
 \backslash maketag@@@ 29, 29, 921, 934
 \backslash mapsto 444
 \backslash mapstochar 444, 445
 \backslash math@ 282, 355, 366
 \backslash math@bgroup 598
 \backslash math@cr 27, 28, 50, 68, 68, 885, 894, 1241, 1478, 2100, 2316, 2341
 \backslash math@cr@ 27, 28, 886, 890, 891
 \backslash math@cr@@ 28, 891, 892
 \backslash math@cr@@@ 23, 23, 28, 28, 38, 63, 892, 895, 1157, 1160, 1255, 1291, 1508, 1596, 1604, 2349
 \backslash math@cr@@@align 1508, 1520
 \backslash math@cr@@@align@measure 1535, 1596
 \backslash math@cr@@@aligned 39, 1141, 1160, 1179
 \backslash math@cr@@@alignedat 1157, 1179
 \backslash math@cr@@@gather 41, 1255, 1307
 \backslash math@egroup 19, 598
 \backslash mathaccent 16, 17, 17, 17, 18, 536, 584, 643, 645
 \backslash mathaccentV 16, 16, 17, 17, 17, 18, 531, 573, 577, 601
 \backslash mathalpha 556
 \backslash mathbin 666
 \backslash mathbin@ 293, 294, 295, 367
 \backslash mathch@ 286, 357
 \backslash mathchar 184, 265
 \backslash mathchardef 493, 610, 624, 697, 698, 700, 701
 \backslash mathchoice 18, 19, 19, 23, 238, 455, 456, 483, 485
 \backslash mathclose 128, 142
 \backslash mathcode 493, 697, 698, 700, 701
 \backslash mathdisplay 2486, 2614, 2623
 \backslash mathdisplay@pop 2495, 2609
 \backslash mathdisplay@push 2494, 2594
 \backslash mathdisplay@pop 2495, 2608, 2617, 2626
 \backslash mathdisplay@push 2494, 2593, 2612, 2621
 \backslash mathdisplay@stack 2592, 2603, 2605, 2609
 \backslash mathellipsis 10, 332, 334
 \backslash mathgroup 599, 607, 608, 615, 617, 618
 \backslash mathindent 3, 3, 3, 45
 \backslash mathinner 376
 \backslash mathit 15
 \backslash mathop 13, 21, 21, 450, 484, 525, 528, 678, 680, 688, 736
 \backslash mathopen 126, 140
 \backslash mathord 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258
 \backslash mathpalette 605, 693, 703, 720, 722, 724, 729, 731, 733
 \backslash mathpunct 272
 \backslash mathrel 9, 450, 703, 704, 736
 \backslash mathrel@ 298, 299, 300, 369
 \backslash mathring 556, 558
 \backslash mathsm@sh 693, 703
 \backslash Mathstrut@ 488
 \backslash Mathstrutbox@ 488, 517, 518
 \backslash mathsurround 599
matrix environment 24, 24, 773
 \backslash matrix 24, 24, 774
 \backslash matrix@check 24, 774, 782, 798, 822
 \backslash matrix@error 784, 788
 \backslash maxcol@width 47, 1584, 1895, 1899, 1903, 1990, 1999
 \backslash maxcolumn@widths 47, 47, 1583, 1585, 1623, 1631
 \backslash maxfields@ 49, 49, 1005, 1154, 1155, 1156, 1180, 1413, 1414, 1419, 1524, 1540, 1548, 1606, 1610, 1611, 1614, 1649, 1664
 \backslash mb@b 691
 \backslash mb@t 691
 \backslash mb@tb 691
 \backslash mdots@ 337, 346
 \backslash mdots@@ 346, 347, 374
 \backslash meaning 186, 353, 399, 495, 534
 \backslash meaning@ 283, 287, 305, 353, 354, 355, 357, 358, 361, 362, 366, 367, 369, 399, 400, 401
 \backslash meaning@@ 354, 359
 \backslash measure@ 49, 1513, 1587
 \backslash measuring@false 1182, 1543, 2348
 \backslash measuring@true 1274, 1589, 2346
 \backslash medmuskip 106, 108, 666, 668, 706
 \backslash medspace 107
 \backslash MessageBreak 153, 155, 567, 882, 1445, 2219
 \backslash min@alignsep 39, 48, 50, 50, 52, 54, 55, 59, 1021, 1161, 1638, 1675, 1676, 1702, 1734, 1735

- 1781, 1800,
1801, 1845,
1867, 1868,
1936, 1961, 1962
- `\mintagsep` . 34, 48,
48, 49, 53, 55,
73, 1020, 1316,
1716, 1783,
1797, 1847,
1849, 1864,
1938, 1940,
1955, 1973,
2557, 2575, 2578
- `\mkern` ... 245, 246,
273, 455, 457,
458, 459, 483,
485, 666, 668,
670, 671, 672,
673, 707, 708,
709, 739, 740,
745, 746, 747,
748, 816, 817, 819
- `\mmeasure@` 66, 2255, 2344
- `\mod` 20, 672
- `\moveleft` ... 2133,
2142, 2181, 2186
- `\mskip` 101, 112, 243,
272, 273, 666, 668
- `\mspace` 112
- `\multicolumn` 815
- `\MultiIntegral` 461,
462, 463, 464, 465
- `\multilimits@` 23
- `\multiply` ... 1156,
1338, 1419,
1782, 1790,
1846, 1937, 1948
- multiline environment
..... 33, 35,
65, 66, 69, 69, 2241
- multiline* environ-
ment 2241
- `\multiline@` . 2237, 2248
- `\multiline@indent` .
.. 2276, 2279,
2294, 2305, 2308
- `\multlinegap` ... 65,
65, 65, 67,
2221, 2279,
2315, 2324,
2338, 2404,
2412, 2418,
2433, 2439, 2450
- `\multlinetaggap` 65,
65, 2221, 2298,
2299, 2334,
2385, 2415, 2436
- `\multlinetaggap@` .. 65
- N
- `\NeedsTeXFormat` 1
- `\negmedspace` 108
- `\negthickspace` ... 111
- `\negthinspace` 105
- `\new@ifnextchar` 27,
779, 827, 864, 891
- `\newbox` 488, 500
- `\newcount` . 98, 216,
217, 288, 307,
571, 781, 855,
1000, 1001, 1005
- `\newcounter` 832
- `\newdimen` 519, 1017,
1018, 1019,
1022, 1023, 1024
- `\newhelp` . 29, 911, 1014
- `\newif` 2, 9,
26, 27, 37, 405,
451, 987, 988,
996, 997, 998,
999, 1406, 1408
- `\newmuskip` 813
- `\newskip` 39, 2221, 2223
- `\next@` . 31, 31, 219,
220, 221, 224,
225, 226, 231,
232, 233, 235,
267, 270, 296,
301, 310, 312,
320, 326, 345,
411, 420, 421,
426, 427, 433,
969, 971, 1142,
1143, 1145,
1234, 1236,
1440, 2237, 2239
- `\nextii@`
219, 222, 294,
296, 299, 301,
315, 316, 323,
325, 409, 411, 413
- `\nextiii@` ... 220, 228
- `\nextiv@` 222, 223
- `\nextix@` 232, 234
- `\nextv@` .. 223, 224, 225
- `\nextvii@` 225, 227
- `\nextviii@` ... 228, 229
- `\nextviii@` 230, 231, 232
- `\noaccents@` 521, 522, 523
- `\noalign` ... 28, 28,
718, 727, 893,
901, 1054, 1071,
1279, 1597,
2144, 2570, 2589
- `\nobreakdash` ... 9, 266
- `\nogood@displaybreak`
..... 864
- `\noindent` ... 907, 1077
- `\nointerlineskip` .
... 718, 727, 1074
- `\nolimits` 7,
13, 20, 22, 23, 686
- `\nomath@env`
. 1228, 1431, 2227
- `\nonmatherr@` .. 591,
664, 1149, 1214
- `\nonscript` 272, 666, 668
- `\nonumber` 31, 32, 32,
32, 968, 977,
978, 1308, 1483,
1521, 1538,
2092, 2233, 2474
- `\normalbaselines` .
... 903, 1365,
1383, 2009,
2023, 2286, 2327
- `\normalcolor` . 925, 928
- `\normalfont` ... 526,
529, 922, 925, 928
- `\normallineskiplimit`
... 504, 505, 1059
- `\not` 350
- `\not@` 322, 323, 324, 361
- `\notag` 32, 977
- `\null` 38,
765, 1152, 1216,
1384, 2024, 2328
- `\nulldelimiterspace`
..... 513, 676
- `\number`
4, 6, 15, 32, 32,
119, 121, 134,
136, 290, 364,
624, 1011, 1111
- `\numberwithin` 847
- O
- `\of` 236
- `\oint` 454
- `\ointop` 454
- `\omit` 1525, 1536
- `\openup` 26, 1048
- `\operator@font` ...
... 667, 671, 673
- `\or` ... 47, 50, 182,
290, 365, 862,
871, 1031, 1032,
1043, 1044,
1097, 1567,
1571, 1625,
1631, 1648, 1663
- `\over` . 6, 6, 13, 145, 160
- `\overarrow@`
717, 720, 722, 724
- `\overfullrule` 35
- `\overleftarrow` ... 721
- `\overleftrightharrow` 723

- `\overline` .. 9, 260, 261
`\overrightarrow` .. 719
`\overset` 21, 677
`\overwithdelims` 148, 163
- P**
- `\p@equation` 936
`\PackageError` 90
`\PackageWarning` ..
 152, 2218
`\PackageWarningNoLine`
 539
`\par` 3, 24, 37
`\parshape` 905
`\partial` 19
`\PassOptionsToPackage`
 23, 25
`\place@tag` 60, 1529, 2004
`\place@tag@gather`
 32, 33, 1267, 1360
`plain.tex` 11, 24, 25, 35
`\plainroot@` 221, 226,
 227, 233, 234, 236
`pmatrix` environment 24
`\pmatrix` 24, 798
`\pmb` 4
`\pmod` 20, 671
`\pod` 20, 669, 671
`\postdisplaypenalty`
 902, 2520
`\predisplaypenalty` 908
`\prevdepth` . 1073, 1078
`\primfrac`
 151, 160, 161,
 162, 163, 164, 165
`\print@eqnum`
 950, 982, 985, 993
`\process@envbody` .
 1120,
 1124, 1125, 1139
`\ProcessOptions` . 3, 54
`\prod` 200, 201
`\prod@` 200, 201
`\protect` 5, 26
`\protected@edef` .. 838
`\ProvidesPackage` ... 3
`\push@begins`
 ... 37, 1127, 1131
- Q**
- `\quad` 830
- R**
- `\r@t` 238, 239, 241
`\raise` 245
`\raise@tag` 32, 32, 35,
 974, 976, 1068,
 1280, 1368,
 1382, 1598,
 2012, 2022,
 2254, 2292,
 2326, 2570, 2589
`\raisetag` .. 32, 32, 973
`\rangle` 385
`\rbrace` 384, 806
`\rbrack` 382
`\rceil` 386
`\ref` 854
`\refstepcounter` ..
 . 26, 837, 986, 994
`\relax` ... 5, 27, 28, 65
`\relaxnext@`
 218, 293, 298,
 314, 322, 339, 406
`\Relbar` .. 437, 441,
 704, 714, 715, 716
`\relbar` .. 435, 439,
 703, 711, 712, 713
`\rendmultline@` ...
 ... 67, 2244, 2320
`\rendsplits@` . 2083, 2116
`\renewenvironment`
 24, 25
`\repeat` . 1632, 1725,
 1757, 1822, 1919
`\RequirePackage` ..
 87, 88, 89
`\reset@equation` 992,
 2087, 2467, 2474
`\reset@strutbox@` .
 14, 35,
 502, 506, 507, 1047
`\resetMathstrut@` . 488
`\restore@hfuzz` 2485,
 2497, 2500,
 2539, 2559, 2582
`\restore@math@cr` 28,
 754, 766, 895,
 1218, 2106, 2252
`\restorealignstate@`
 1095, 1481
`\restorecolumn@` ..
 1109, 1201
`\restorecounters@`
 1082,
 1294, 1609, 2397
`\rfloor` 387
`\rgroup` 388
`\rhook` 447
`\RIfM@` 1148,
 1213, 1227, 2226
`\right` 390,
 514, 800, 802,
 806, 808, 809, 824
`\Rightarrow`
 437, 443, 715, 716
`\rightarrow` 435, 442,
 444, 446, 712, 713
`\Rightarrowfill@` . 715
`\rightarrowfill@` .
 712,
 720, 729, 734, 750
`\rightdelim@` 11, 379, 397
`\rmoustache` 389
`\romannumeral` ... 1011
`\root` 218
`\rootbox` 236, 245
`\row@` 42,
 42, 63, 1000,
 1100, 1259,
 1312, 1317,
 1514, 1523,
 1537, 1600,
 1718, 1719,
 1724, 1738,
 1750, 1751,
 1756, 1764,
 1777, 1815,
 1816, 1821,
 1829, 1842,
 1912, 1913,
 1918, 1926,
 1932, 2007,
 2020, 2141,
 2250, 2352, 2358
`\rq` 67
`\rVert` ... 136, 141, 809
`\rvert` ... 121, 127, 808
- S**
- `\savealignstate@` .
 1095, 1425
`\savecolumn@` 1109, 1151
`\savecounters@` 1082,
 1277, 1594, 2351
`\savefieldlength@`
 . 1563, 2038, 2042
`\saveshift@`
 .. 1041, 1722,
 1739, 1741,
 1754, 1786,
 1788, 1819,
 1853, 1855,
 1916, 1944, 1946
`\savetaglength@` ..
 63, 1025,
 1288, 1559, 2140
`Sb` environment 23
`\scriptfont` 755, 756, 757
`\scriptscriptstyle`
 . 23, 182, 237, 239
`\scriptstyle`
 182, 239, 739,
 740, 760, 768, 769
`\select@group` 578
`\set@field` 61,
 2039, 2043, 2047
`\set@fontsize` 14

- `\set@gather@field` 1264, 1360
`\set@mathaccent` .. 193, 195, 197, 199, 201, 203, 205, 207, 209, 211
`\setb@ck` 30
`\setbox` 95, 236, 243, 489, 492, 503, 576, 635, 636, 639, 640, 683, 684, 737, 738, 1278, 1384, 1595, 1618, 1619, 1627, 1628, 2024, 2105, 2118, 2120, 2123, 2126, 2128, 2136, 2165, 2167, 2169, 2171, 2174, 2176, 2296, 2328, 2353, 2429, 2448, 2499, 2530, 2556, 2577
`\setboxz@h` ... 241, 269, 1262, 1266, 1283, 1287, 1528, 1551, 2037, 2041, 2139, 2356, 2374, 2428, 2447
`\setcounter` 839, 840, 844
`\setlength` 65
`\shdots@for` .. 812, 814
`\shift@tag` 1025, 2007, 2020
`\shifftag@false` .. 1257, 1370, 1389, 2394
`\shifftag@true` 1324, 1330, 1342, 2392
`\shoveleft` 44, 69, 69, 2350, 2400
`\shoveright` 44, 69, 69, 2350, 2400
showkeys package .. 71, 71, 71
`\sideset` 21, 681
`\sixt@n` 865
`\size@update` ... 14, 14
`\SK@label` . 2465, 2477
`\SK@equationtrue` . 2465, 2491
`\SK@tagform@` 2476, 2477
`\skewchar` ... 617, 621
`\skip@` 973, 974, 2563, 2568, 2576, 2579
`\slimits@` 7, 21, 22, 189, 191, 193, 195, 197, 199, 201, 203, 205, 207, 209, 211
smallmatrix environ- ment ... 23, 765
`\smash` 21, 690
Sp environment 23
`\span` . 1268, 1517, 1602
split environment 42, 61, 61, 62, 63, 63, 63, 63, 2069
`\split` 1251, 1436
`\split@` 2131, 2137, 2152, 2154, 2162, 2179, 2184, 2191, 2193, 2196
`\split@al@a` . 2200, 2201
`\split@al@tagcheck` 2205, 2209
`\split@aligned` ... 2072, 2198
`\split@err` . 2060, 2076
`\split@tag` .. 2085, 2090, 2094, 2098
`\split@warning` ... 2202, 2217, 2500
`\spread@equation` . 14, 32, 991, 1047, 1052, 1165, 1219, 2096, 2565, 2586
`\sqrtsign` 8, 8, 240, 241
`\st@rredfalse` .. 44, 1239, 1451, 1461, 1476, 1496, 2242, 2613
`\st@rredtrue` ... 44, 1246, 1456, 1466, 1471, 1491, 1501, 2247, 2474, 2622
`\start@align` 44, 44, 1411, 1451, 1456, 1461, 1466, 1471, 1476, 1491, 1496, 1501
`\start@aligned` . 38, 1147, 1204, 1211
`\start@gather` 1226, 1239, 1246
`\start@multiline` .. 2225, 2242, 2247
`\std@equal` 698, 701, 704
`\std@minus` 697, 700, 703
`\stepcounter` 36, 1553, 2376
`\strut` 14, 20, 675
`\strut@` 14, 500, 1077, 1170, 1221, 1261, 1266, 1287, 1528, 1551, 2036, 2110, 2269, 2288, 2331
`\strutbox` 505
`\strutbox@` 35, 500, 1075, 1076
subarray environment 23, 23, 23, 23, 752
`\subarray` 764
`\substack` 23, 23, 23, 764
`\sum` 202, 203
`\sum@` 202, 203
- T
- `\tabskip` 50, 50, 50, 58, 58, 67, 1172, 1176, 1258, 1265, 1268, 1515, 1601, 2040, 2045, 2257, 2565, 2586
`\tag` 23, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 30, 30, 30, 30, 30, 31, 31, 31, 31, 31, 31, 32, 33, 38, 38, 66, 66, 73, 918, 919, 965, 1252, 1509, 2107, 2253, 2256, 2472, 2491
`\tag*` 29, 29
`\tag@false` 1068, 1279, 1597, 2254, 2493, 2510, 2600
`\tag@help` . 29, 911, 917
`\tag@in@align` 71, 962, 1252, 1509, 2253
`\tag@in@display` .. 2469, 2491
`\tag@in@display@a` 2469, 2470
`\tag@lengths` 1026, 1032, 1101, 1276, 1592
`\tag@shifts` .. 1025, 1044, 1713

- 1747, 1812, 1909
`\tag@true` 967,
 1522, 1539,
 2092, 2371,
 2372, 2474, 2600
`\tag@width` .. 1025,
 1317, 1719,
 1738, 1751,
 1777, 1816,
 1842, 1913, 1932
`\tagform@` 26,
 29, 30, 30, 30,
 854, 921, 926,
 928, 930, 935,
 985, 993, 1554,
 2377, 2477, 2514
`\tagshift@` 52, 1019,
 1690, 1691,
 1694, 1697,
 1700, 2006, 2019
`\tagsleft@false` ..
 29, 53, 55
`\tagsleft@true`
 28, 52, 57
`\tagwidth@`
 . 58, 59, 1022,
 1317, 1320,
 1321, 1337,
 1340, 1344,
 1345, 1777,
 1780, 1792,
 1842, 1844,
 1859, 1932,
 1935, 1951,
 1975, 2382,
 2384, 2414, 2435
`\tbinom` 6, 171
`tbtags` option 65
`\tdots@` 339, 426
`\text` 4
`\textellipsis` 334, 337
`\textfont` 34,
 494, 617, 621, 1020
`\textstyle`
 . 23, 182, 238, 450
`\textup` 26, 854
`\tfrac` 6, 6, 168
`\thedots@`
 290, 291, 347,
 348, 349, 350,
 352, 361, 364,
 365, 368, 370, 372
`\theequation` . . . 26,
 838, 841, 926,
 928, 985, 993,
 1554, 2377, 2514
`\theparentequation`
 26, 838, 841
`\thetag` 30, 930
`\thickmuskip`
 109, 111, 706
`\thickspace` .. 110, 768
`\thinmuskip`
 102, 104, 273, 706
`\thinspace` 103
`\thr@@` 469,
 757, 1614, 1628,
 1630, 1631,
 2128, 2132,
 2133, 2142,
 2146, 2158,
 2176, 2180,
 2181, 2185, 2186
`\Tilde` 20, 656
`\tilde` 20, 547, 656
`\tmspace` 5,
 100, 102, 104,
 106, 108, 109, 111
`\too@wide`
 . 2524, 2527, 2540
`\totwidth@` 50, 50, 52,
 58, 1023, 1099,
 1241, 1275,
 1284, 1285,
 1296, 1299,
 1300, 1375,
 1376, 1478,
 1621, 1622,
 1635, 1680,
 1690, 1703,
 1704, 1705,
 1732, 1933,
 2100, 2258,
 2260, 2263,
 2359, 2363, 2383
`\try@load@fontshape` 81
`\tw@` .. 319, 403, 468,
 510, 525, 528,
 636, 637, 684,
 685, 738, 743,
 755, 756, 1020,
 1156, 1338,
 1349, 1412,
 1416, 1419,
 1471, 1496,
 1501, 1613,
 1644, 1650,
 1665, 1682,
 1693, 1771,
 1796, 1836,
 1848, 1863,
 1872, 1939,
 1954, 1966,
 1972, 1995,
 2126, 2136,
 2174, 2556,
 2560, 2577, 2583
`\typeout` 56, 66

U

`\uccode` 280,
 285, 292, 297,
 302, 308, 313, 321
`\underarrow@`
 725, 729, 731, 733
`\underleftarrow` .. 730
`\underleftrightarrow`
 732
`\underrightarrow` . 728
`\underset` 21, 679
`\unhbox` 269,
 743, 1628,
 2136, 2541,
 2551, 2569, 2590
`\unhcopy` 2124,
 2172, 2557, 2580
`\unpenalty` 1619
`\unskip` 339,
 923, 1628, 2121,
 2125, 2127,
 2136, 2173,
 2175, 2541,
 2551, 2557,
 2569, 2580, 2590
`\unvbox` 1619
`\unvcopy` 2119, 2165, 2168
`\uproot` 215,
 219, 222, 232, 234
`\uproot@` 216,
 222, 234, 235, 243
`\use@mathgroup` . . . 578
`\usefont` 94

V

`\value` 839, 844
`\varDelta` 249
`\varGamma` 248
`\varLambda` 251
`\varOmega` 258
`\varPhi` 256
`\varPi` 253
`\varPsi` 257
`\varSigma` 254
`\varTheta` 250
`\varUpsilon` 255
`\varXi` 252
`\vbox` 23, 23, 23, 23
`\vcenter` 23,
 23, 23, 514,
 753, 765, 1153,
 1217, 1425,
 2097, 2133,
 2152, 2181, 2191
`\Vec` 663
`\vec` 554, 663
`\veqno` 2462, 2463, 2516
`\Vert` 5, 131
`\vert` 5, 116

- `\vtop` 63,
 726, 1153, 1217,
 1381, 2021,
 2118, 2165, 2325
- W**
- `\wd` 244, 637,
 640, 642, 687,
 743, 1621, 1631,
 2132, 2133,
 2142, 2180,
 2181, 2185,
 2186, 2297,
 2441, 2451, 2575
- `\wdz@` ... 1032, 1284,
 1285, 1328,
 1336, 1339,
 1353, 1575,
 2359, 2363,
 2382, 2442, 2452
- `\widehat` 16, 561
- `\widetilde` ... 16, 560
- `\write` 31
- X**
- `\xc@calc@shift@lc` .
 ... 60, 1914, 1922
- `\xc@calc@shift@lf` .
 53, 53,
 53, 54, 1720, 1728
- `\xc@calc@shift@rc` .
 1817, 1825
- `\xc@calc@shift@rf` .
 ... 54, 1752, 1760
- `\x@lcalc@width` ...
 ... 60, 1929, 1985
- `\x@rcalc@width` . 54,
 1767, 1832, 1890
- `xalignat` environment
 .. 44, 44, 51, 1450
- `xalignat*` environ-
 ment 1450
- `\xatlevel@` . 44, 44,
 48, 1406, 1412,
 1416, 1613,
 1615, 1636,
 1693, 1796,
 1848, 1863,
 1939, 1954, 1972
- `\xi` 19
- `\xleaders` 817
- `\xleftarrow` ... 21, 751
- `\xrightarrow` 21, 734, 750
- `xxalignat` environ-
 ment ... 44,
 44, 48, 49, 51, 1450
- `\xxat@true` 1417
- Z**
- `\z@` 6, 6, 13, 28