The eqnlines Package

Niklas Beisert

Institut für Theoretische Physik
Eidgenössische Technische Hochschule Zürich
Wolfgang-Pauli-Strasse 27, 8093 Zürich, Switzerland

nbeisert@itp.phys.ethz.ch

2025/04/30, v0.8

https://ctan.org/pkg/eqnlines
https://github.com/nbeisert/latex-pkg-nb

Abstract

eqnlines is a LaTeX 2ε package providing a framework for typesetting single- and multi-line equations which extends the established equation environments of LaTeX and the amsmath package with many options for convenient adjustment of the intended layout. In particular, the package adds flexible schemes for numbering, horizontal alignment and semi-automatic punctuation, and it improves upon the horizontal and vertical spacing options. The extensions can be used and adjusted through optional arguments and modifiers to the equation environments as well as global settings.

Contents

1 Introduction

Typesetting mathematical equations is an undisputed strength of TEX. LATEX improved the overall management of display equations, for instance by providing optional numbering. It also added elementary functionality for multi-line equations with alignment. Some of its deficiencies were addressed by the multi-line equation environments of the package amsmath which have become an established standard for these purposes.

The package eqnlines builds upon and extends the functionality of the LATEX and amsmath equation environments with some new features as well as convenient options to adjust the layout where needed. The main additions are as follows:

- Equation numbers can be assigned to individual lines (as for `align` and `gather`) or once for the multi-line equation block (as for `multline`). In the former case, a sub-numbering scheme can be applied (as through `subequations`). In the latter case, the position can be assigned to a specific line (first/middle/last/chosen). Moreover, equation numbers can be turned on and off by commands, and they can be triggered by setting a label.
- The vertical spacing above and below single- and multi-line equations of LaTeX and amsmath can be somewhat variable, hard to control and even resistive in certain situations. The package implements clearer structures controlling the vertical spacing, including proper dependency on the text line above and ways to adjust the spacing.
- The framework introduces a scheme which semi-automatically inserts punctuation, e.g. '.' or ',', at the end of the following (or every) equation environment. Punctuation can also be inserted at every alignment column or equation line including the possibility to prepend a certain spacing.
- Next to `\[...\]` as an alias for the single-line `equation` environment, the package uses `\<...\>` as an alias multi-line equations.
- The horizontal alignment and indentation of equation lines can be adjusted via a scheme or on a line-by-line basis.
- The alignment marker can be placed before or after the equation signs while maintaining proper spacing to symbols before and after it. This simplifies the construction of continuing equations in an aligned context.
- Equation lines are subject to shrinking of space if the available space does not suffice (analogously to single-line equations).
- Most settings can be controlled via optional arguments and modifiers to the equation environment or via global settings. This includes switching between different types of equation environments, enabling or disabling numbering, adjusting vertical spacing, etc. This feature simplifies the adjustment and fine-tuning of equations towards the intended layout.
- Last but not least, the underlying amsmath code, originating from the TeX era and early LaTeX years, has been redesigned with emphasis on clarity, readability, adjustability and maintainability (but at the cost of moderately higher resource consumption and moderately lower efficiency). Nevertheless, it remains original LaTeX 2_ε code without using the expl3 layer.

The package represents a stand-alone implementation of an equations environment which is largely compatible with the established LaTeX and amsmath environments `equation`, `multline`, `gather`, `align` and their variants. Hence, the package can be used instead of amsmath with no or minor modifications to the LaTeX sources for single- and multi-line equations. It can also be used alongside amsmath including the mathtools extensions to make use of the additional maths typesetting features provided by these packages. In the latter case, the equation environments of LaTeX and amsmath are either replaced or left in place while the eqnlines environments can be accessed using the alternate name `equations`.

2 Usage

Notice regarding package version v0.8: Please note that this package is still in a development and testing stage in the present version. This mainly applies to the documentation of features and code: Currently, the documentation is basic and minimal without extensive coverage of all features and settings, and it lacks desirable illustrations and examples.

It is likely that some features of the package do not work to full extent, and that the package will not cooperate well with other packages. Therefore, please report any malfunctions that you may notice.

Therefore, it is likely that internal macros and mechanisms will change, It is also conceivable that the public interface will change in minor but relevant ways in order to accommodate for important adjustments or additional features. It is intended that such changes would only require minor adaption of document sources that use an early version of this package.

To use the eqnlines package add the command

```
\usepackage{eqnlines}
```

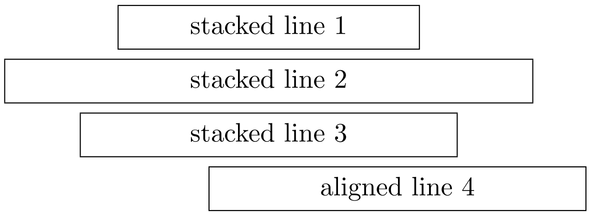
to the preamble of the LaTeX document. To use unrelated features of the amsmath package or of the mathtools extension, it makes sense to load these packages *before* eqnlines.

2.1 Equations Environment

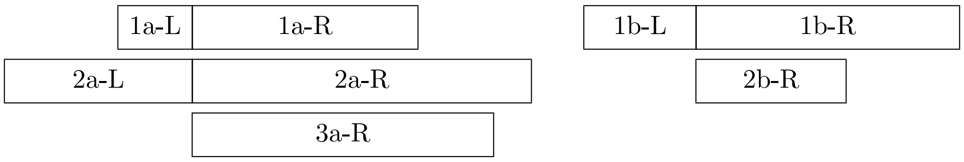
`equations` (*env.*) The package supplies a main maths environment called `equations` which has three principal modes of operation. It can display a single-line equation just as the LaTeX environment `equation` or the symbolic shortcut `\[...\]`:

single line

It can display a stack of equations analogous to the amsmath environments `gather` and `multline`:[1]



It can also display several columns of aligned equations analogous to the amsmath environment family `align`:



The environment `equations` accepts a comma-separated list of optional parameters '`[`*opts*`]`':

```
\begin{equations}[opts]␣
...
\end{equations}
```

Furthermore, the environment accepts some modifiers (like the star modifier '`*`' for many other LaTeX macros) which will be explained further below. These follow the scheme `{ !t~ !t* !t! !o !e{@} }` according to the syntax of `\NewDocumentCommand`.

[1]Arguably, a single-line equation is just a stack of equations of height 1. Nevertheless, there is a single-line mode which prohibits line breaks and which works slightly more efficiently: For example, the multi-line modes will process the input twice which is not needed for the single-line mode. Apart from that, the package takes care that the layout and spacing of single-line equations and multi-line equations consisting of a single line is the same.

We note that the equations environment should be started with a whitespace character '␣' which provides a clear separation from optional arguments '`[`*opts*`]`' and/or modifiers which must immediately follow the environment declaration `\begin{equations}` without whitespaces.

`single` (*key*) `lines` (*key*) `columns` (*key*) The three modes of operation are selected by setting an optional argument as follows:

purpose	single-line equation	stacked equation(s)	aligned equations
name	`single`	`lines`	`columns`
alt. names	`equation`, `eq`, `1`	`gather`, `ga`, `ln`, `~`	`align`, `al`, `col`, `@`
symbolic	`\[␣...\]`	`\<~␣...\>`	`\<␣...\>`
`amsmath` env.	`equation`	`gather`, `multline`	`align`
columns	—	single	multiple, aligned
alignment	adjustable	adjustable	alternating right/left
parsing	single, direct	two passes	two passes
numbering	on/off	off/single/multiple	off/single/multiple

The aligned mode more or less encompasses all three modes, and the stacked mode with only a single line is more or less just a single equation. However, the more complex forms also come along with some restrictions, hence, it makes sense to use the appropriate mode for the intended equation content. For instance, a single equation simply reads the equation input once, while the multi-line equation environments parse the environment body twice which can potentially disrupt some other functionality that is included in the body. Furthermore, the horizontal adjustment options are very restricted in aligned mode, and therefore the aligned form can automatically reduce to the stacked form (with right alignment) if only a single column is provided (no '`&`'s).

```
\begin{equations}[single]
x=\cos\phi
\end{equations}
```

$$x = \cos\phi \qquad (1)$$

```
\begin{equations}[lines]
x=\cos\phi \\ \phi=\arccos x
\end{equations}
```

$$x = \cos\phi \qquad (2)$$
$$\phi = \arccos x \qquad (3)$$

```
\begin{equations}[columns]
x&=\cos\phi & \phi&=\arccos x \\
 &=(z+z^{-1})/2 & &=-i\log z
\end{equations}
```

$$\begin{aligned} x &= \cos\phi & \phi &= \arccos x & (4)\\ &= (z+z^{-1})/2 & &= -i\log z & (5)\end{aligned}$$

`\[...\]` `\<...\>` `~` (*key*) `sqropt` (*key*) `angopt` (*key*) The package offers several alternative names for the same mode as well as a symbolic short form `\<...\>` extending the LaTeX display equation form `\[...\]` to multi-line equations. Here, the tilde '`~`' in `\<~␣...\>` is a modifier character which acts as a short form for the optional argument `lines` selecting the lines mode. Both short forms can be customised by setting default arguments via the global options `sqropt={`*opts*`}` and `angopt={`*opts*`}`. Both default arguments are preset to `nonumber` which disables equation numbering, see section 2.2.

```
\[
x=\cos\phi
\]
```

$$x = \cos\phi$$

```
\<~
x=\cos\phi \\ \phi=\arccos x
\>
```

$$x = \cos\phi$$
$$\phi = \arccos x$$

```
\<
x&=\cos\phi & \phi&=\arccos x \\
 &=(z+z^{-1})/2 & &=-i\log z
\>
```

$$\begin{aligned} x &= \cos\phi & \phi &= \arccos x\\ &= (z+z^{-1})/2 & &= -i\log z\end{aligned}$$

```
\eqnlinesset{sqropt={donumber}}
\[ x=\cos\phi \]
```

$$x=\cos\phi \tag{6}$$

`equation` (*env.*) `gather` (*env.*) `multline` (*env.*) `align` (*env.*)

The package also supplies or overwrites the `amsmath` environments `equation`, `gather`, `multline`, `align` and `flalign` including their starred at `-at` variants (but not the `split` construction). It is possible to define further equation environments *env* with a predefined set of options *opts* using:

`\[re]newenvironment{`*env*`}{\eqnaddopt{`*opts*`}\equations}{\endequations}`

```
\begin{equation}
x=\cos\phi
\end{equation}
```

$$x=\cos\phi \tag{7}$$

```
\begin{gather}
x=\cos\phi \\ \phi=\arccos x
\end{gather}
```

$$x=\cos\phi \tag{8}$$
$$\phi=\arccos x \tag{9}$$

```
\begin{align}
x&=\cos\phi & \phi&=\arccos x \\
 &=(z+z^{-1})/2 & &=-i\log z
\end{align}
```

$$\begin{aligned} x&=\cos\phi & \phi&=\arccos x && (10)\\ &=(z+z^{-1})/2 & &=-i\log z && (11)\end{aligned}$$

```
\newenvironment{eqnlist}
  {\eqnaddopt{lines,shape=left}\equations}
  {\endequations}
\begin{eqnlist}[nonumber]
x=\cos\phi \\ \phi=\arccos x
\end{eqnlist}
```

$$\begin{aligned}&x=\cos\phi\\&\phi=\arccos x\end{aligned}$$

2.2 Numbering

`numberline` (*key*) `nline,n` (*key*)

The package extends the established interface of LaTeX and the `amsmath` package for labelling equations with numbers or with manually assigned tags. For multi-line equations, there are two distinct modes of operations: individual labelling of the equation lines or one overall number/tag for the whole block of equations. The modes are selected by an optional argument `numberline=`*mode* (alternatively `nline` or just `n`) as follows:

name	alt.	description	preset
`none`	`n`	individual lines	all lines, preset off
`all`	`a`		all lines
`sub`	`s`		subequations (a, b, c, . . .)
`first`	`f`	single number	first line
`last`	`l`		last line
`middle`	`m`		middle line
`out`	`o`		last/first line for right/left tags
`in`	`i`		first/last line for right/left tags
`here`	`h`		line indicated by `\numberhere`
`best`	`*`		line with most available space

```
\begin{equations}[!,numberline=...]
  x  &= \cos\phi  \\ &= (z+z^{-1})/2 \\
\phi &= \arccos x \\ &= -i\log z
\end{equations}
```

`none`:

$$\begin{aligned} x &= \cos\phi \\ &= (z+z^{-1})/2 \\ \phi &= \arccos x \\ &= -i\log z \end{aligned}$$

`all`:

$$\begin{align} x &= \cos\phi \tag{12}\\ &= (z+z^{-1})/2 \tag{13}\\ \phi &= \arccos x \tag{14}\\ &= -i\log z \tag{15}\end{align}$$

`sub`:

$$\begin{align} x &= \cos\phi \tag{16a}\\ &= (z+z^{-1})/2 \tag{16b}\\ \phi &= \arccos x \tag{16c}\\ &= -i\log z \tag{16d}\end{align}$$

`first`:

$$\begin{align} x &= \cos\phi \tag{17}\\ &= (z+z^{-1})/2 \notag\\ \phi &= \arccos x \notag\\ &= -i\log z \notag\end{align}$$

`middle`:

$$\begin{align} x &= \cos\phi \notag\\ &= (z+z^{-1})/2 \tag{18}\\ \phi &= \arccos x \notag\\ &= -i\log z \notag\end{align}$$

`last`:

$$\begin{align} x &= \cos\phi \notag\\ &= (z+z^{-1})/2 \notag\\ \phi &= \arccos x \notag\\ &= -i\log z \tag{19}\end{align}$$

`best`:

$$\begin{align} x &= \cos\phi \tag{20}\\ &= (z+z^{-1})/2 \notag\\ \phi &= \arccos x \notag\\ &= -i\log z \notag\end{align}$$

`bestlineauto` (*key*) Note that the mode `best` (line with most available space) is activated automatically if the (single) tagged line does not have sufficient space to hold the tag. This feature can be controlled by the setting `bestlineauto=`*bool*.

`\nonumber`
`\donumber` Numbering can be turned on and off (for individual lines or for the block as a whole depending on the mode) by means of:

`\nonumber` and `\donumber`

`nonumber` (*key*)
`donumber` (*key*)
`number` (*key*)
`nn,*` (*key*)
`dn,!` (*key*) The numbering can be disabled or enabled for the block by the keys `nonumber` or `donumber` (`nn`=‘`*`’ or `dn`=‘`!`’ for short) or by `number=`*bool* with *bool* either `on` or `off` (among several alternative forms). Alternatively the number can be switched by using modifiers (which cannot be used in conjunction with optional arguments `[...]`):

`\[*␣...\]` and `\[!␣...\]`

This allows to define a default behaviour and specify exceptions where they may occur. The star modifier following directly the environment declaration replaces the starred form of environments (`equation*`, etc.) and there is no need to adjust the closing statement.

`\numberhere`
`\numbernext` The placement of a single number for an equation block can be adjusted by:

`\numberhere` and `\numbernext`

The former macro overrides the position to the present line, the latter macro defers the number to the next line. For example, if an equation is broken into several lines one may use the combination `\numbernext \\` to assign the number to the last line.

```
\begin{equations}
   x &= \cos\phi \nonumber \\
     &= (z+z^{-1})/2 \\
\phi &= \arccos x \nonumber \\
     &= -i\log z
\end{equations}
```

$$\begin{align} x &= \cos\phi \notag\\ &= (z+z^{-1})/2 \tag{21}\\ \phi &= \arccos x \notag\\ &= -i\log z \tag{22}\end{align}$$

```
\begin{equations}*
   x &= \cos\phi \donumber \\
     &= (z+z^{-1})/2 \\
\phi &= \arccos x \donumber \\
     &= -i\log z
\end{equations}
```

$$\begin{align} x &= \cos\phi \tag{23}\\ &= (z+z^{-1})/2 \notag\\ \phi &= \arccos x \tag{24}\\ &= -i\log z \notag\end{align}$$

```
\eqnlinesset{numberline=last}
\<! x &= \cos\phi \\
 \phi &= \arccos x \>
```

$$\begin{align} x &= \cos\phi \notag\\ \phi &= \arccos x \tag{25}\end{align}$$

```
\eqnlinesset{angopt=donumber}
\<* x &= \cos\phi \\
 \phi &= \arccos x \>
```

$$\begin{aligned} x &= \cos\phi \\ \phi &= \arccos x \end{aligned}$$

```
\begin{equations}
   x &= \cos\phi \numbernext \\
     &= (z+z^{-1})/2 \\
\phi &= \arccos x \numbernext \\
     &= -i\log z
\end{equations}
```

$$\begin{aligned} x &= \cos\phi \\ &= (z+z^{-1})/2 \qquad (26) \\ \phi &= \arccos x \\ &= -i\log z \qquad (27) \end{aligned}$$

```
\eqnlinesset{numberline=here}
\<!
   x &= \cos\phi \\
     &= (z+z^{-1})/2 \\
\phi &= \arccos x \numberhere \\
     &= -i\log z
\>
```

$$\begin{aligned} x &= \cos\phi \\ &= (z+z^{-1})/2 \\ \phi &= \arccos x \qquad (28) \\ &= -i\log z \end{aligned}$$

```
\eqnlinesset{numberline=first}
\<!
   x &= \cos\phi \numbernext \\
     &= (z+z^{-1})/2 \\
\phi &= \arccos x \numbernext \\
     &= -i\log z
\>
```

$$\begin{aligned} x &= \cos\phi \\ &= (z+z^{-1})/2 \qquad (29) \\ \phi &= \arccos x \\ &= -i\log z \end{aligned}$$

`\label` `\tag` Equation numbers can receive LaTeX labels as usual and they can be turned into manually assigned tags using the established macros:

`\label[`*name*`]{`*label*`}` and `\tag[*]{`*tag*`}`

The optional parameter *name* for `\label` assigns a name to the label which can be referenced by `\nameref`. A `tag` replaces the equation number, `tag*` will drop the decoration by parentheses. Note that a label and a tag will always apply to the next number that will be printed, and only a single label and/or tag may be specified for it. For example, if the present line has no numbering, but the following line does, `\label` or `\tag` will apply to the following line. The macros `\label` and `\tag` can also be instructed to automatically enable numbering/tagging for the present line or block via `\donumber`, see below. By default, numbering/tagging is triggered for `\tag`, but not for `\label` reflecting the behaviour set forth by `amsmath`. By enabling triggering for `\label`, numbers will be produced only if they have a chance of being referenced.

`label` (*key*) `tag` (*key*) `labelname` (*key*) `@` (*key*) The equations environment provides an alternative means to specify labels and tags within the optional arguments `[`*opts*`]` or via the modifier `@{`*label*`}` (which may *follow* further optional arguments):

`label={`*label*`}`, `tag[*]={`*tag*`}`, `labelname={`*name*`}`, `\[@{`*label*`}`…`\]`

In particular, in subequations mode (`sub`), the optional argument `label` can be used to assign a label to the parent number addressing the whole equation block.

`\eqref` The macro `\eqref` is the standard method for referring to equation numbers via their label. This method also uses the layout defined below.

`\eqref{`*label*`}`.

`\tagform` `\tagbox` `\tagboxed` For custom typesetting, `\tagform` encloses a number/tag with decoration, `\tagbox` puts the decorated number in a box and `\tagboxed` combines the two.

`tagbox` (*key*) `tagform` (*key*) The typesetting of equation numbers and tags passes through two macros, one which defines the layout and another one which adds a decoration by parentheses. These two methods can be adjusted via the options:

`tagbox[*]={`*code*`}` and `tagform={`*l*`{`*code*`}`*r*`}` or `tagform*={`*code*`}`

Here, *code* is some macro code that references the argument '`#1`' containing the number or tag, and *l* and *r* can be opening and closing parentheses for the tag presentation.

```
\eqnlinesset{tagform=[{#1}]}
\eqnlinesset{tagbox={\textcolor{blue}{#1}}}
\<[!,numberline=last]
   x &= \cos\phi \\
     &= (z+z^{-1})/2 \\
\phi &= \arccos x \\
     &= -i\log z
\>
```

$$\begin{aligned} x &= \cos\phi \\ &= (z+z^{-1})/2 \\ \phi &= \arccos x \\ &= -i\log z \qquad [30] \end{aligned}$$

2.3 Horizontal Adjustment

`layout` (*key*) `center` (*key*) `left` (*key*) First of all, the overall layout can be adjusted between central and left alignment via `layout=center`, `layout=left` or `center`, `left` for short.

```
\<[layout=center]
   x &= \cos\phi \\
     &= (z+z^{-1})/2 \\
\phi &= \arccos x \\
     &= -i\log z
\>
```

$$\begin{aligned} x &= \cos\phi \\ &= (z+z^{-1})/2 \\ \phi &= \arccos x \\ &= -i\log z \end{aligned}$$

```
\<[layout=left]
   x &= \cos\phi \\
     &= (z+z^{-1})/2 \\
\phi &= \arccos x \\
     &= -i\log z
\>
```

$$\begin{aligned} x &= \cos\phi \\ &= (z+z^{-1})/2 \\ \phi &= \arccos x \\ &= -i\log z \end{aligned}$$

`tags` (*key*) `tagsright` (*key*) `tagsleft` (*key*) Furthermore, numbers and/or tags may be placed on the right or left margin via `tags=right`, `tags=left` or `tagsright`, `tagsleft` for short.

```
\<[tags=right,!]
   x &= \cos\phi \\
     &= (z+z^{-1})/2 \\
\phi &= \arccos x \\
     &= -i\log z
\>
```

$$\begin{aligned} x &= \cos\phi && (31) \\ &= (z+z^{-1})/2 && (32) \\ \phi &= \arccos x && (33) \\ &= -i\log z && (34) \end{aligned}$$

```
\<[tags=left,!]
   x &= \cos\phi \\
     &= (z+z^{-1})/2 \\
\phi &= \arccos x \\
     &= -i\log z
\>
```

$$\begin{aligned} (35) && x &= \cos\phi \\ (36) && &= (z+z^{-1})/2 \\ (37) && \phi &= \arccos x \\ (38) && &= -i\log z \end{aligned}$$

`tagmargin` (*key*)
`tagmargin*` (*key*)
`tagmarginratio` (*key*)

In central alignment layout, one can impose a tag margin `tagmargin={`*dimen*`}` which allocates some space to the tag such that equation content is centred in the remaining horizontal space. The margin can also be set to the width of some text by `tagmargin*={`*text*`}` or it can be calculated as the maximum width of tags by `tagmargin` without parameter (default). The option `tagmarginratio={`*ratio*`}` uses the tag margin only for equation blocks with a ratio of tags to rows above the given (decimal) ratio (a value above 1 uses the tag margin only for single equations with tags; default is `0.334`). The option `tagmarginthreshold={`*threshold*`}` uses the tag margin only if the ratio of spacings would be below the given (decimal) threshold (very much off balance; default is `0.5`). The latter two options together with some tag margin can produce a more appealing layout for equation blocks of mixed filling. In the following example, the former two equations are centred on all horizontal space while the latter two equations are centred on the space left of the tag (the ratio of spacings without tag margin would be very small here):

```
\eqnlinesset{tagmarginthreshold=0.7}
\[! \framebox[4em]{} \]
\[! \framebox[8em]{} \]
\[! \framebox[12em]{} \]
\[! \framebox[16em]{} \]
```

$$\boxed{\hspace{4em}} \tag{39}$$
$$\boxed{\hspace{8em}} \tag{40}$$
$$\boxed{\hspace{12em}} \tag{41}$$
$$\boxed{\hspace{16em}} \tag{42}$$

`leftmargin` (*key*)
`leftmargin*` (*key*)
`minleftmargin` (*key*)
`maxleftmargin` (*key*)

In left alignment layout, all equations are left aligned to a left margin (`leftmargin` is initialised to the first level of enumerations and itemisations). It can be set to the width of some text by `leftmargin*={`*text*`}`. Depending on the situation, the left margin may be reduced or extended to `minleftmargin` or `maxleftmargin`, respectively.

```
\eqnlinesset{layout=left}
\<
   x &= \cos\phi \\
     &= (z+z^{-1})/2 \\
\phi &= \arccos x \\
     &= -i\log z
\>
\<[tags=left,!]
   x &= \cos\phi \\
     &= (z+z^{-1})/2 \\
\phi &= \arccos x \\
     &= -i\log z
\>
```

$$\begin{aligned} x &= \cos\phi \\ &= (z+z^{-1})/2 \\ \phi &= \arccos x \\ &= -i\log z \end{aligned}$$

$$\begin{aligned} (43)\quad x &= \cos\phi \\ (44)\quad &= (z+z^{-1})/2 \\ (45)\quad \phi &= \arccos x \\ (46)\quad &= -i\log z \end{aligned}$$

`fulllength` (*key*)
`mincolsep` (*key*)
`maxcolsep` (*key*)

The horizontal alignment of columns is fixed for aligned multi-line equations: Each pair of subsequent columns forms a unit which is aligned at the intermediate alignment marker '`&`'. These columns are distributed evenly over the available horizontal space. Here, the outer space left and right of the set of columns is treated on equal footing to the space between the columns (option `fulllength=off`; default), but it can be eliminated so that the outer columns are pushed right to the margin (option `fulllength=on`). A minimum and maximum column separation can be specified via `mincolsep=`*dimen* and `maxcolsep=`*dimen* (defaults are `2em` and `1em`) or the maximum column separation can be disabled by `maxcolsep=off` (which is implied by `fulllength=on`).

```
\<[maxcolsep=2em]
 x &= \cos\phi     & \phi &= \arccos x \\
   &= (z+z^{-1})/2 &      &= -i\log z \>
```

$$\begin{aligned} x &= \cos\phi & \phi &= \arccos x \\ &= (z+z^{-1})/2 & &= -i\log z \end{aligned}$$

```
\<[maxcolsep=off]
 x &= \cos\phi     & \phi &= \arccos x \\
   &= (z+z^{-1})/2 &      &= -i\log z \>
```

$$\begin{aligned} x &= \cos\phi & \phi &= \arccos x \\ &= (z+z^{-1})/2 & &= -i\log z \end{aligned}$$

```
\<[fulllength]
 x &= \cos\phi     & \phi &= \arccos x \\
   &= (z+z^{-1})/2 &      &= -i\log z \>
```

$$\begin{aligned} x &= \cos\phi & \phi &= \arccos x \\ &= (z+z^{-1})/2 & &= -i\log z \end{aligned}$$

For stacks of equations including single equations, there is just a single alignment column whose horizontal alignment can be adjusted via a shape scheme or by manually adjusting individual lines. A shape scheme determines the horizontal alignment for each line and it is specified by the optional argument `shape=`*mode* as follows:

name	alt.	shape	alignment
`default`	`def`	uniform	default
`left`	`l`		left
`center`	`c`	uniform	central
`right`	`r`		right
`first`	`indent`, `rc`	first/rest	first line indented
`hanging`	`outdent`, `lc`	first/rest	first line hanging
`steps`	`lcr`	first/intermediate/last	left/centre...centre/right

Note that the `steps` shape comes to use in the `amsmath` environment `multline`.

```
\eqnlinesset{pad=2em}
\<~[shape=...] x = \cos\phi \\ x = (z+z^{-1})/2 \\
  \phi = \arccos x \\ \phi = -i\log z \>
```

`left`:

$$x = \cos\phi \\ x = (z+z^{-1})/2 \\ \phi = \arccos x \\ \phi = -i\log z$$

`center`:

$$x = \cos\phi \\ x = (z+z^{-1})/2 \\ \phi = \arccos x \\ \phi = -i\log z$$

`right`:

$$x = \cos\phi \\ x = (z+z^{-1})/2 \\ \phi = \arccos x \\ \phi = -i\log z$$

`first`:

$$x = \cos\phi \\ x = (z+z^{-1})/2 \\ \phi = \arccos x \\ \phi = -i\log z$$

`hanging`:

$$x = \cos\phi \\ x = (z+z^{-1})/2 \\ \phi = \arccos x \\ \phi = -i\log z$$

`steps`:

$$x = \cos\phi \\ x = (z+z^{-1})/2 \\ \phi = \arccos x \\ \phi = -i\log z$$

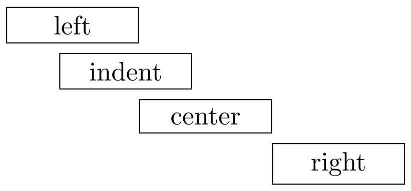
`\shoveleft` `\shovecenter` `\shoveright` The alignment preset can be adjusted for individual lines by the macros:

`\shoveleft|center|right`[`*|!|`[*dimen*]],

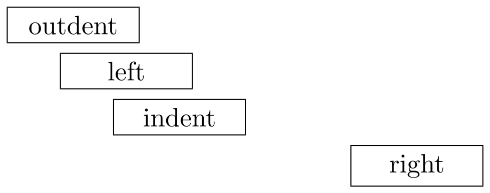
In contradistinction to `amsmath`, these macros do not require to specify the cell contents as their argument (but there is no harm in doing so). The macro accept an optional argument `indent` (*key*) [*dimen*] specifying a variable amount of shift. They also accept the modifiers '`*`' or '`!`' for indentation or hanging indentation by the standard indentation amount (`indent=2em`). `\shoveby` Furthermore, `\shoveby`[`*`]{*dimen*} shifts the line by the amount *dimen*, where the star variant resets the shift amount first.

`padding` (*key*) `padleft` (*key*) `padright` (*key*) Note that (hanging) indentation requires to add some padding around the equations block via the optional argument `padding|padleft|padright`[=`{`*dimen*`}`]. The value `indent` sets the padding to the default indentation amount and `max` extends the padding to all available space. Note that `indent*={`*dimen*`}` sets the default indentation amount and the left padding at the same time.

```
\eqnlinesset{indent=2em,pad=5em}
\<~
\shoveleft   \framebox[5em]{left} \\
\shoveleft*  \framebox[5em]{indent} \\
\shovecenter \framebox[5em]{center} \\
\shoveright  \framebox[5em]{right}
\>
```



```
\eqnlinesset{layout=left}
\eqnlinesset{leftmargin=2em}
\eqnlinesset{indent=2em}
\<~
\shoveleft! \framebox[5em]{outdent} \\
\shoveleft  \framebox[5em]{left} \\
\shoveleft* \framebox[5em]{indent} \\
\shoveright \framebox[5em]{right}
\>
```



`alignshrink` (*key*) `tagshrink` (*key*) `alignbadness` (*key*) `tagbadness` (*key*) Finally, we note that within single and stacked equations, very long equations that do not fit the available horizontal space are subject to shrinking attempts. In other words, TEX will attempt to shrink the glue contained in the equation line to make it fit. This shrinking can be controlled by the two parameters `alignbadness` and `tagbadness` accepting integer values. They are used towards determining whether to shift away from the intended alignment or whether to raise or lower the equation tag, respectively. Small values prevent shrinking and higher values allow for more compression.

```
\<~!
x+x \\
x+x+x+x \\
x+x+x+x+x+x \\
x+x+x+x+x+x+x+x \\
x+x+x+x+x+x+x+x+x+x \\
x+x+x+x+x+x+x+x+x+x+x+x \\
\>
```

$$x+x \tag{47}$$
$$x+x+x+x \tag{48}$$
$$x+x+x+x+x+x \tag{49}$$
$$x+x+x+x+x+x+x+x \tag{50}$$
$$x{+}x{+}x{+}x{+}x{+}x{+}x{+}x{+}x{+}x \tag{51}$$
$$x{+}x{+}x{+}x{+}x{+}x{+}x{+}x{+}x{+}x{+}x{+}x \tag{52}$$

`mintagsep` (*key*) `\raisetag` If the available space on a line does not suffice to place both the equation and its tag (with a minimum separation of `mintagsep`; default is `0.5em`), a tag will automatically be lowered or raised (depending on whether it is placed on the right or left). The macro `\raisetag` may be used to fine-tune the vertical placement (applies only if the tag is already shifted due to lack of space).

```
\[! \phi = -\int \frac{\mathrm{d}x}{\sqrt{1+x^2}} \]
```

$$\phi = -\int \frac{\mathrm{d}x}{\sqrt{1+x^2}} \tag{53}$$

```
\[! x = \frac{\partial}{\partial \phi}\sin\phi
    \raisetag{0.45\baselineskip} \]
```

$$x = \frac{\partial}{\partial\phi}\sin\phi \tag{54}$$

`margin` (*key*) `marginleft` (*key*) `marginright` (*key*) `linewidth` (*key*)

The margins and line width of an equation block can be adjusted by `margin`, `marginleft`, `marginright` or `linewidth`. The equations and corresponding numbers or tags will be fit within these bounds. This feature can be used within lists or enumerations to undo an indentation.

```
\[ \indicate{line width} \]

\[[margin=2em] \indicate{reduced} \]

\begin{itemize}
\item first level
  \[ \indicate{default width} \]
  \[[marginleft=0pt]
    \indicate{full width} \]
\end{itemize}
```

$$\boxed{\text{line width}}$$

$$\boxed{\text{reduced}}$$

- first level

$$\boxed{\text{default width}}$$

$$\boxed{\text{full width}}$$

2.4 Punctuation

Extending proper punctuation across equations is a delicate matter, and maintaining it while redacting the text certainly takes more attention to detail than many author are willing to afford. A contributing factor is that punctuation marks are harder to spot alongside equation context and somewhat out of place anyway.

`\eqnpunctdefault` `\eqnpunct` `punct` (*key*)

The package supplies a semi-automatic scheme by which equations are terminated by a specific punctuation mark.[2] Punctuation marks are set by:

```
\eqnpunctdefault{punct}   \eqnlinesset{punct={punct}}
\eqnpunct{punct}          \eqnaddopt{punct={punct}}
\[[punct={punct}] ...\]
```

The former two forms set and enable a default punctuation mark; the middle two forms set the punctuation mark for the next equation environment in line; the final form applies to the equation environment only. For example, one might declare '`\eqnpunctdefault.`' to terminate all equations with a period '`.`'. The default behaviour can be adjusted to a comma '`,`' for an individual equation by declaring '`\eqnpunct,`' before the equation (i.e. at the end of the textual phrase to which the punctuation mark belongs), at the end of the equation or by using the optional argument `[punct={,}]`. Likewise, `\eqnpunct{}` and `[punct{}]` eliminate a preset punctuation.

```
\eqnpunctdefault.
The equation
\[ x = \cos\phi \eqnpunct{} \]
can also be written as
\eqnpunct,
\[ x = (z+z^{-1})/2 \]
where we assume
\[ z = \exp(i\phi) \]
```

The equation

$$x = \cos\phi$$

can also be written as

$$x = (z + z^{-1})/2,$$

where we assume

$$z = \exp(i\phi).$$

`\eqnpunctapply` In situations, where the punctuation must appear before the end of the block, e.g. before a "q.e.d.", it can be invoked manually by `\eqnpunctapply`.

`punctsep` (*key*) For convenience, one may also specify a desired space (or any other code sequence) preceding

[2]Clearly, the implementation of the scheme will takes higher efforts than direct coding. Hence, the scheme can be useful in situations where equations typically terminate phrases or where punctuation is otherwise expected in regular patterns.

the punctuation by `[punctsep={`*sep*`}]`, e.g. *sep*=`\,` or *sep*=`\␣`.

`\eqnpunctcol` `\eqnpunctline` `punctcol` (*key*) `punctline` (*key*) For multi-line equations, there are two further levels of default punctuation for terminating columns and lines which are specified via the macros `\eqnpunctcol` and `\eqnpunctline` or the optional arguments `punctcol` and `punctline`. A punctuation item may also be handed on to the next lower level of punctuation via the starred forms `punct*` and `punctline*`.

```
\eqnpunctcol, \eqnpunctline;
\eqnpunctdefault.
\< x &= \cos\phi &
\phi &= \arccos x \\
   x &= (z+z^{-1})/2 &
\phi &= -i\log z \>
```

$$\begin{aligned} x &= \cos\phi, & \phi &= \arccos x; \\ x &= (z+z^{-1})/2, & \phi &= -i\log z. \end{aligned}$$

2.5 Math Classes at Alignment

Alignment in multi-line equations breaks equations into components before and after the alignment position. Unfortunately, this also interrupts TEX's math spacing mechanism which is based on the math classes assigned to the characters, and there appears to be no direct way of determining the math class to the previous letter. Therefore, one has to make some assumptions on the letters that will surround the alignment marker '`&`' in order to obtain the appropriate spacing also across the alignment.

The `amsmath` environment `align` assumes that the left column ends with an ordinary character. This leads to the correct spacing when an equation $a = b + c$ is broken before the equals relation as `a&=b+c`, and also if an equation sequence continues on the next line as `\\&=d-e`. However, it is difficult to achieve the right spacing if the right-hand side is to be broken into several lines: For instance, `\\&␣+f` aligns the subordinate binary operation with the equals sign (which may be undesirable). Instead placing a phantom equals sign is an effort that somewhat disrupts the readability of the code.

`class` (*key*) `ampeq` (*key*) `eqamp` (*key*) The package implements a more flexible assignment of math classes at the alignment. The above default behaviour is invoked by the optional argument `class=ampeq` (or `ampeq` for short). The optional argument `class=eqamp` (or `eqamp` for short) imposes math classes at the alignment such that an equation sign should be placed just before the alignment. Concretely, it inserts `\mathrel{}` classes just before and after the alignment marker. Furthermore, in case of an empty left alignment cell, the leading math class is changed to `\mathord{}` so that a following binary operator is not interpreted as a unary one. For example, the following two expressions produce (almost) identical output:

```
\<[class=ampeq]
a &= b+c \\
  &= d-e \\
  &\mathrel{}\phantom{=} +f
\>
```

$$\begin{aligned} a &= b+c \\ &= d-e \\ &\mathrel{} +f \end{aligned}$$

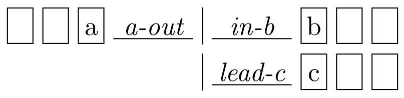
```
\<[class=eqamp]
a =& b+c \\
  =& d-e \\
   & +f
\>
```

$$\begin{aligned} a &= b+c \\ &= d-e \\ &\mathrel{} +f \end{aligned}$$

`classout` (*key*) `classin` (*key*) `classlead` (*key*) Math classes just before and after alignment can be adjusted freely by the optional arguments:

`classout={`*class*`}`, `classin={`*class*`}`, `classlead={`*class*`}`.

The parameter `classlead` alternatively `classin*` determines the math class just after the alignment if the cell before alignment is empty. The spacing at the alignment is determined by the pairing of the last/first character and the selected math class at the alignment:



2.6 Vertical Spacing

Display equations in TeX are considered to be part of the surrounding text. Hence, the vertical spacing depends on the surrounding text, in particular on the width and depth of the last line of text. Due to this influence it can be difficult to manually adjust the spacing accurately. The package adds several options to control the vertical spacing, and it also implements a uniform behaviour for all types of equations.

The spacing of equations to the surrounding text is a combination of several aspects:

First, TeX inserts some interline spacing according to its rules. The amount depends on the depth/height of the surrounding text and the height/depth of the math content. The former typically takes rather uniform values, whereas the latter can range wildly with the context (plain equations vs. fractions and matrices). As equations are normally surrounded by a relatively large amount of glue, it makes sense to reduce the dependency on the height/depth of math content. Therefore, the package makes equation environments appear to the surrounding text as a line with a fixed height and depth, and thus interline glue merely fills some potential gaps of the surrounding text. The apparent height and depth are defined by `displayheight` and `displaydepth` which default to the dimensions of a strut.

`displayheight` (*key*)
`displaydepth` (*key*)

Second, the spacing of display equations depends on the width of the previous line of text. If the math content fits well into the available horizontal space, the display equation is called short and less glue is needed above the equation. The package implements this basic TeX feature for all single- and multi-line equation environments.

```
example of a long text line:
\[ \mbox{long mode} \]
vs.\ short:
\[ \mbox{short mode} \]
following line
```

example of a long text line:

long mode

vs. short:

short mode

following line

`shortmode` (*key*)

TeX also reduces the amount of glue below short equations (potentially to make their spacing appear more uniform). The package allows to adjust the spacing for short equations via the global option `shortmode=`*mode* where *mode* takes the values:

mode	reduced glue
`off`	disabled
`above`	above short equations (package default)
`belowone`	also below short single-line equations
`belowall`	also below all short multi-line equations

`short` (*key*)
`long` (*key*)

Short and long amounts of glue can also be enforced for individual equation environments via the optional arguments `short` and `long` taking the values `above`, `below` or `both`.

```
example of a long text line:
\[[short] \mbox{forced short} \]
and short:
\[[long] \mbox{forced long} \]
following line
```

example of a long text line:

forced short

and short:

forced long

following line

There are three special situations `cont`, `par` and `top` which trigger different spacings: `cont` describes the situation at the start of an empty horizontal list (invoked by `\noindent`) or when an equation block directly follows another one; here, the space above the equation should be minimal (or even negative to remove the space below the previous equation block). `par` describes the situation at the beginning of a paragraph (invoked by `\par`); here, the space above the equation adds to the space between paragraphs. `top` describes the situation at the top of a vertical list (invoked by `\nointerlineskip`); here, one would typically want no space.

```
\hrule\begin{minipage}{\linewidth}
\[ \mbox{top} \]
some text\par
\[ \mbox{par} \]
\[ \mbox{cont} \]
\end{minipage}\hrule
```

top

some text

par

cont

Third, the package provides several means to adjust the glue around equations:

`noskip` (*key*) `medskip` (*key*) Next to `short` and `long` the spacing above and below equation environments can be reduced to some other fixed smaller amount via `medskip` or removed altogether via `noskip`. These keys also take the values `above`, `below` or `both`.

```
\hrule
\[[long] \mbox{long default} \]
\hrule
\[[medskip] \mbox{medium space} \]
\hrule
\[[noskip] \mbox{no space} \]
\hrule
```

long default

medium space

no space

`par` (*key*) The key `par` controls whether the equation environments end in horizontal mode (value `cont`) or in vertical mode (value `par`, default) with a dedicated amount of glue `belowparskip`. An environment can also be made to end in vertical mode without interline skip (value `top`) using the glue `belowtopskip`.

`...skip` (*key*) `\vspace` `...space` (*key*) Variable amounts of skip can be set via `aboveskip` and `belowskip` or `skip` for both simultaneously. In addition, the package extends the `\vspace` mechanism of LaTeX to equation bodies where it adds vertical space below the next equation line or below the equation environment. Additional glue can be added above or below equation environments by means of the options `abovespace` and `belowspace`.

`...skip` (*key*) The package also maintains several global vertical space settings `above`*pos*`skip` and `below`*pos*`skip` (sometimes *pos*`skip` for both):

...*pos*`skip`	both	description
`...long...`	`longskip`	regular amount of glue
`...short...`	–	reduced glue for short equations
`...cont...`	–	glue when issued from an empty `\noindent` paragraph
`...par...`	–	glue when starting a paragraph (in vertical mode)
`...top...`	–	glue when issued at the top of vertical list
`...med...`	`medskip`	medium amount of glue
`...tag...`	`tagskip`	glue for outer raised/lowered tags
`...medtag...`	`medtagskip`	glue for outer raised/lowered tags with medium glue
`...partag...`	–	glue for outer raised/lowered tags with par skip

`...mode` (*key*) The situations *pos*=`cont`, `par` and `top` use the respective amount of glue `above`*pos*`skip` above the equations and the regular amount of glue `belowlongskip` below. These behaviours may be adjusted by the global options `above`*pos*`mode` and `below`*pos*`mode` with the values:

value	reduced glue
`long`	regular amount of glue
`short`	reduced glue for short equations
`cont`	amount for empty paragraph
`par`	amount for paragraph (and end the paragraph)
`top`	amount for top (and end the paragraph without interline skip)
`noskip`	no glue
`medskip`	medium amount of glue

`spread` (*key*) `strut` (*key*) Likewise, the spacing between the lines of a multi-line equation environment can be adjusted via `spread={`*dimen*`}` which defaults to `\jot≡3pt`. In addition, all equation lines and tags are supplied with struts to ensure a minimum height and depth. The latter behaviour is controlled by the switch `strut`.

`prebreak` (*key*) `postbreak` (*key*) `allowbreaks` (*key*) `prepenalty` (*key*) `postpenalty` (*key*) `interpenalty` (*key*) `\displaybreak` Finally, the breaking of multi-line equations across pages can be controlled as follows: The setting `allowbreaks` (or `allowdisplaybreaks`) taking values `0` (never) through `4` (permissive) controls the permittivity of page breaks within multi-line equations. The optional arguments `prebreak` and `postbreak` taking values `0` (do not) through `4` (enforce) suggest a break just above or below the equation environment. The command `\displaybreak[`*val*`]` with values `0` through `4` (default) suggests a break below the current line or below the equation environment.

2.7 Further Environments and Features

The package supplies some additional environments and features:

`equationsbox` (*env.*) `\<...\>` **Equation Boxes.** The package provides a boxed equation environment `equationsbox` which can be used within arbitrary math content. It works analogously to `equations` including optional arguments and modifiers, but it offers a reduced range of functionality such as (evidently) no numbering (yet, the `lines` mode accepts multiple columns here). It can also be invoked by the symbolic short form `\<...\>` when called within math mode.

`top,t` (*key*) `center,c` (*key*) `bottom,b` (*key*) `margin` (*key*) `marginleft` (*key*) `marginright` (*key*) `colsep` (*key*) `frame` (*key*) `wrap` (*key*) The equations box accepts several arguments: `top`, `center`, `bottom` (or `t`, `c`, `b`) specify the vertical alignment of the box. `margin`, `marginleft`, `marginright` specify additional margin space around the equations box. `colsep` specifies the amount of separation between the columns. `frame`[=*cmd*] encloses the equations box by a *cmd* such as `\fbox` which accepts one argument (or a command sequence which ends with a macro accepting one argument). `wrap={{`*cmdl*`}{`*cmdr*`}}` surrounds the equations box by the two commands *cmdl* and *cmdr*.

```
\[\left\{
\begin{equationsbox}[margin=1em]
   x &= \cos\phi \\
\phi &= \arccos x
\end{equationsbox}
\right\}\]
```

$$\left\{\quad\begin{aligned} x &= \cos\phi \\ \phi &= \arccos x \end{aligned}\quad\right\}$$

```
$\Longrightarrow\<~[shape=l,frame]
   x = \cos\phi &
\phi = \arccos x \\
   x = (z+z^{-1})/2 &
\phi = -i\log z
\>\Longleftarrow$
```

$\Longrightarrow \boxed{\begin{array}{ll} x = \cos\phi & \phi = \arccos x \\ x = (z+z^{-1})/2 & \phi = -i\log z \end{array}} \Longleftarrow$

`subequations` (*env.*) `subeqtemplate` (*key*) **Collective Numbering.** The environment `subequations` groups equations contained in the body with a common primary equation number and an extra level of numbering (typically: a, b, c, . . .). The numbering layout can be controlled via `subeqtemplate`. For instance, the default behaviour of adding lowercase latin letters to the parent equation number (`#1`) is achieved by:

```
subeqtemplate={#1\alph{#2}}
```

```
\eqnlinesset
  {subeqtemplate={#1-\roman{#2}}}
\begin{subequations}
\[! x = \cos\phi \]
and
\[! \phi = \arccos x \]
\end{subequations}
```

$$x = \cos\phi \tag{55-i}$$

and

$$\phi = \arccos x \tag{55-ii}$$

`intertext` (*env.*) `\intertext` **Text Intermissions.** The environment `intertext` (equivalently the macro `\intertext`) injects a (short) line of text into a multi-line equation while preserving the equation alignment across the text. The intertext environment must replace the end of line marker ‘`\\`’ between two lines of the equation (to avoid blank lines). The environment accepts several of the vertical spacing adjustments as an optional argument.

```
\< x &= \cos\phi
\intertext[medskip]{and}
\phi &= \arccos x \>
```

$$x = \cos\phi$$

and

$$\phi = \arccos x$$

`\framecell` **Framed Cells.** The package allows to frame cells of an equation block via issuing a simple command within the cell:

`\framecell[`*cmd*`]`

This command corresponds to `\Aboxed` of `mathtools`. In particular, when used within columns or aligned mode, the frame will extend over both right and left alignment components of a cell; in order to allocate the right amount of space, it should be issued within the first cell of the pair. The layout of the frame can be adjusted by the optional argument *cmd* which defaults to `\fbox`: it must be a macro which accepts one argument (or a command sequence which ends with a macro accepting one argument). Note: Any semi-automatic punctuation is included within the frame, see section 2.4. Parts of a cell can be framed by the amsmath macro `\boxed`, which will not include semi-automatic punctuation. Furthermore, the height and depth of the box are bounded from below by a strut, see section 2.6.

```
\< x &= \cos\phi \\
\framecell \phi &= \arccos x \>
```

$$x = \cos\phi$$
$$\boxed{\phi = \arccos x}$$

```
\[\framecell[\fboxrule2pt\fbox]
  \mbox{important} \eqnpunct! \]
```

$$\boxed{\text{important!}}$$

```
\[\framecell[\fcolorbox{white}{yellow}]
  \mbox{highlight}\]
```

highlight

alt (*key*) \eqnalt **Alternative Content Description.** The package provides a basic interface to describe the equation content in an alternative form for the purposes of accessibility or documentation (corresponding to the `alt` tag in html):

`alt={`*alt text*`}` or `\eqnalt[`*opt*`]{`*alt*`}`

At the moment the alternative text *alt* is not processed further, but an accessibility extension may implement the feature in tagged pdfs or html conversion. The comma-separated optional arguments *opt* may specify the content further: `line` and `cell` restrict the applicability to the current equation line or cell, respectively. Other keys might specify the content format and language.

```
\<[alt={example equations}]
x &= \cos\phi \\
\eqnalt[line]{reverse relationship}
\phi &= \arccos x \>
```

$$x = \cos\phi$$
$$\phi = \arccos x$$

2.8 General Options

\eqnlinesset Options of general nature can be selected by the commands:

```
   \usepackage[opts]{eqnlines}
or \PassOptionsToPackage{opts}{eqnlines}
or \eqnlinesset{opts}
```

`\PassOptionsToPackage` must be used before `\usepackage`; `\eqnlinesset` must be used afterwards. *opts* is a comma-separated list of options.

The package supplies the following general settings:

option	description
`defaults=classic`	mimic classic LaTeX/amsmath (layout and dimensions)
`defaults=eqnlines`	eqnlines layout with fontsize-relative dimensions
`rescan`	rescan environment body for special commands (e.g. `\verb`)
`linesfallback`	single column in align mode reverts to lines mode
	value `reuse` avoids third measuring pass
`ampproof`	equip optional argument parsing with protection for ‘`&`’
`crerror`	invoke an error when ‘`\\`’ is used in a single equation
`scanpar`	allow scanning of `\par` within equation body
	(e.g., for use in nested `\parbox` or `minipage`)

2.9 Feature Selection and Package Options

The following few settings can only be specified when loading the package, not via `\eqnlinesset`:

option	type	description
`equation`	bool	provide/overwrite `equation` and `\[...\]`
`amsmath`	bool	provide/overwrite `amsmath` environments and macros
`amsmathends`	bool	patch `amsmath` environments with individual endings
`backup`	bool	backup original `amsmath` environments as `ams...`
`ang`	bool	provide `\<...\>`
`eqref`	bool	provide `\eqref`

If the above settings are explicitly disabled, the package will only supply the general purpose environment `equations` and its boxed cousin `equationsbox`. In that case, the specific equation environments and other features can be activated by the command:

`\eqnlinesprovide{`*features*`}`

features is a comma-separated list of features:

feature	description
env	provide/overwrite environment *env*: `equation`, `gather`, `multline`, `align`, `flalign` `multlined`, `gathered`, `aligned`, `subequations`
env=*name*	provide environment *env* as *name*
`sqr`	provide `\[...\]`
`ang`	provide `\<...\>`
`eqref`	provide/overwrite macro `eqref`
`tagform`	provide/overwrite macro `\tagform@`
`maketag`	provide/overwrite macro `\maketag@@@`

3 Information

3.1 Copyright

3.2 Credits

This package is based on the LaTeX package amsmath (initially named amstex) which in turn is based on the TeX macro system amstex written by Michael Spivak. The initial work of porting amstex to LaTeX was done in 1988–1989 by Frank Mittelbach and Rainer Schöpf. In 1994 David M. Jones added the support for flush-left layout and did extensive improvements to the align 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. Since 2016, the package has been maintained by the LaTeX Project with contributions by the above and David Carlisle.

This package has been forked from amsmath in accordance with the LPPL, particularly paragraph 6. The original package amsmath is available at CTAN within latex-amsmath. It uses the basic mechanisms for processing numbered multi-line equations as developed in amsmath (environments `equation`, `align`, `gather`, `multline`, `gathered`, `aligned` and related), as well as code implementing these mechanisms. It differs from amsmath in the following aspects:

- The implementations of `split` and methods unrelated to multi-line equations and equation numbering have been dropped.
- Code has been restructured, macros have been renamed and extended.
- Numbering and horizontal adjustment schemes have been unified and extended.
- Options for math classes surrounding the alignment have been added.
- A punctuation scheme has been added.
- Vertical spacing has been redesigned.
- Optional parameters have been added to environments.
- Various configuration options and layout settings have been added.
- Cooperation with hyperref, showkeys and amsmath has been included into the package.

3.3 Files and Installation

The package consists of the files:

`README.txt`	readme file
`eqnlines.ins`	installation file
`eqnlines.dtx`	source file
`eqnlines.sty`	package file
`eqnlines-dev.sty`	package file (development version)
`eqnlines.pdf`	manual

The distribution consists of the files `README.txt`, `eqnlines.ins` and `eqnlines.dtx`.

- Run (pdf)LaTeX on `eqnlines.dtx` to compile the manual `eqnlines.pdf` (this file).
- Run LaTeX on `eqnlines.ins` to create the package `eqnlines.sty` and the developers version `eqnlines-dev.sty`. Copy the file `eqnlines.sty` to an appropriate directory of your LaTeX distribution, e.g. *texmf-root*`/tex/latex/eqnlines`.

3.4 Related CTAN Packages

The package is related to other packages available at CTAN:

- This package uses the package keyval to process the options for the package, environments and macros. Compatibility with the keyval package has been tested with v1.15 (2022/05/29).

- This package reproduces the math environments functionality of the package amsmath. The present code is based on amsmath v2.17t (2024/11/05). Compatibility with the amsmath package is maintained whether eqnlines is loaded before or after amsmath. By default, eqnlines overwrites most math environments of amsmath with its own implementations. It can also preserve them as `ams...` if needed. Alternatively, eqnlines may assign individual names to the maths environments and preserve the ones of amsmath. The other features provided by amsmath can be used.
- The package mathtools is a popular extension of the amsmath package. This package incorporates some of the features and improvements provided by the mathtools package. Compatibility with the mathtools package has been tested with v1.31 (2024/10/04), and it is maintained whether eqnlines is loaded before or after mathtools. Some features like emphasising equations via empheq does not (yet) work.
- This package cooperates with the package hyperref to create anchors and references within the electronic document. Compatibility with the hyperref package has been tested with v7.01l (2024/11/05).
- This package supports the display of labels and references through the package showkeys. Compatibility with the showkeys package has been tested with v3.21 (2024/05/23).

3.5 Feature Suggestions

The following is a list of features for consideration towards future versions of this package. Their potential use may range between useful and niche; and their difficulty between easy and impossible:

- expand documentation further
- complete code documentation
- list of all option keys with scope, defaults and special values

3.6 Revision History

v0.8: 2025/04/30

- added framed cells (`\framecell`)
- added automatic best line selection for tag placement (`best` and `bestlineauto`)
- symbolic environment `\<...\>` forwards to `equationsbox` in math mode
- added wrapping for `equationsbox` (`frame`, `wrap`)
- horizontal adjustment reworked and completed; `\shoveby` added
- extended `\label` to assign names to labels for `\namedref`
- interface for alternative representations (`alt` and `\eqnalt`)
- options to adjust line width and margins (`linewidth`, `marginleft`, `marginright`)
- added option `scanpar` to allow `\par` appearing in equation body
- added continuous penalties (`prepenalty`, `postpenalty`, `interpenalty`)
- added overloading for `displaymath` and remaining amsmath math environments
- minor interface changes (rename, recombine, values)
- documentation expanded
- several issues fixed

v0.7.1: 2025/04/09

- improvements for pdf tagging
- backup all available math environments at the start using `backup` switch

v0.7: 2025/04/03

- manual expanded, examples added
- fixes for numbering, tagging, options, `linesfallback`, zero lines
- expansions for vertical spacing modes, tag display, `subeqtemplate`
- some consolidations
- internal rearrangements

v0.6.1: 2025/03/27

- `\eqnpunct` can place punctuation within the current equation cell
- `numberline=none` now acts as `numberline=all` and `nonumber`
- fixed and extended `tagmargin` with `tagmarginratio` and `tagmarginthreshold`
- padding now applies to single-line equations as well

v0.6: 2025/03/11

- preliminary pdf tagging support (`https://latex3.github.io/tagging-project/`; amsmath *must* be loaded *before* eqnlines to avoid errors
- classic LaTeX/amsmath vs. eqnlines presets
- changed vertical spacing schemes and added further options
- supplied dimensions processed by `\glueexpr`
- more independent of amsmath structures
- internal reorganisations

v0.5: 2025/02/25

- preview version published on CTAN

A Implementation

The appendix documents the various components of the present package.

The code for the package is based on the amsmath package, see section 3.1 and section 3.2. It was forked at version v2.17t dated 2024/11/05. Most of the code was substantially redesigned (macros renamed, reshuffled, enhanced), but many of the underlying mechanisms were preserved. The documentation thus contains excerpts from the amsmath package documentation explaining some details of the implementation.

Please note that the documentation is completed only for few sections in the present version. Various open issues are remarked.

B General Support

In the following we describe general purpose supporting routines.

B.1 Development Messages

The package offers a version eqnlines-dev for development and debugging purposes. It outputs extra information on the current location within the code in order to track progress. The extra lines for the development version are indicated as '⟨dev⟩' in the implementation documentation:

```
⟨dev⟩\def\eql@dev#1{\PackageInfo{eqnlines-dev}{#1}}
⟨dev⟩\def\eql@dev@start#1{\eql@dev{starting \string#1}}
⟨dev⟩\def\eql@dev@enter#1{\eql@dev{entering \string#1}}
⟨dev⟩\def\eql@dev@leave#1{\eql@dev{ leaving \string#1}}
⟨dev⟩\def\eql@dev@enterenv{\eql@dev{entering \@currenvir}}
⟨dev⟩\def\eql@dev@leaveenv{\eql@dev{ leaving \@currenvir}}
⟨dev⟩\def\eql@dev@in#1#2{\eql@dev{ \space within \string#1 #2}}
```

B.2 Supporting Definitions

`\eql@false` (*bool*)
`\eql@true` (*bool*)

Rather than the standard LaTeX scheme of `\`*xxx*`false`, `\`*xxx*`true` and `\if`*xxx* for boolean variables *xxx*, we use a scheme where `\`*xxx* is either undefined or defined (to an empty macro) and is tested against by the ε-TeX conditional `\ifdefined\`*xxx*. In order to make the scheme more tangible, we define the two expected values for boolean variables:

```
\let\eql@false\@undefined
\let\eql@true\@empty
```

B.3 Dollardollar Abstraction

`l@dollardollar@begin`
`eql@dollardollar@end`

As of 2025 LaTeX defines `\dollardollar@begin` and `\dollardollar@end` to represent (and adjust) the beginning and end of bare TeX display equations ('`$$`'). For the time being, we make sure to revert to '`$$`' if these macros are not yet available:

```
\ifdefined\dollardollar@begin
  \def\eql@dollardollar@begin{\dollardollar@begin}
  \def\eql@dollardollar@end{\dollardollar@end}
\else
  \def\eql@dollardollar@begin{$$}
  \def\eql@dollardollar@end{$$}
\fi
```

B.4 Look-Ahead in Alignment

Scanning for optional arguments `[...]` or modifiers such as '`*`' using the LaTeX `\@ifnextchar` mechanism has two challenges within aligned equations: a square bracket or star may well be part of the intended mathematical expression and the look-ahead could trip upon an alignment character '`&`' which inadvertently triggers to enter the next alignment column.

`eql@ifnextchar@loose`
`eql@ifnextchar@tight`

To address the first challenge, we can force the special characters to follow immediately the macro invocation. For clarity, we copy LaTeX's original `\@ifnextchar` in `\kernel@ifnextchar` which skips over spaces as `\eql@ifnextchar@loose`. We replicate the amsgen version `\new@ifnextchar` that does not skip over spaces as `\eql@ifnextchar@loose`. The space before `#1` allows to look-ahead for spaces as well:

```
\let\eql@ifnextchar@loose\kernel@ifnextchar
\long\def\eql@ifnextchar@tight#1#2#3{%
  \let\reserved@d= #1%
  \def\reserved@a{#2}%
  \def\reserved@b{#3}%
  \futurelet\@let@token\eql@ifnch@tight
}
\def\eql@ifnch@tight{%
  \ifx\@let@token\reserved@d
    \let\reserved@b\reserved@a
  \fi
  \reserved@b
}
```

`\eql@atxii`

Capture '`@`' as a character (catcode 12) rather than a letter (catcode 11) as `\eql@atxii` so that we can look-ahead for '`@`' with both `\makeatother` and `\makeatletter` modes:

```
\begingroup
  \makeatother
  \let\tmp=@%
  \makeatletter
  \global\let\eql@atxii\tmp
\endgroup
```

`eql@ifnextgobble@...`
`\eql@ifstar@...`
`\eql@testopt@...`
`\eql@teststaropt@...`

We introduce a collection of look-ahead macros which do or do not skip over spaces. The macros `\eql@ifstar@...` and `\eql@testopt@...` replicate the LaTeX counterparts `\@ifstar` and `\@testopt`. The macros `\eql@ifnextgobble@...` work like `\@ifnextchar`, but also gobble the specific character if found; one might define `\eql@ifstar@...` as `\eql@ifnextgobble@...*`. The macros `\eql@teststaropt@...` tests for combinations of '`*`' and optional arguments `[...]`:

```
\long\def\eql@ifnextgobble@loose#1#2{\eql@ifnextchar@loose#1{\@firstoftwo{#2}}}
\long\def\eql@ifnextgobble@tight#1#2{\eql@ifnextchar@tight#1{\@firstoftwo{#2}}}
\long\def\eql@ifstar@loose#1{\eql@ifnextchar@loose*{\@firstoftwo{#1}}}
\long\def\eql@ifstar@tight#1{\eql@ifnextchar@tight*{\@firstoftwo{#1}}}
\long\def\eql@ifat@loose#1#2{\eql@ifnextgobble@loose{@}{#1}{%
  \eql@ifnextgobble@loose\eql@atxii{#1}{#2}}}
\long\def\eql@ifat@tight#1#2{\eql@ifnextgobble@tight{@}{#1}{%
  \eql@ifnextgobble@tight\eql@atxii{#1}{#2}}}
\long\def\eql@testopt@loose#1#2{\eql@ifnextchar@loose[{#1}{#1[{#2}]}}%]
\long\def\eql@testopt@tight#1#2{\eql@ifnextchar@tight[{#1}{#1[{#2}]}}%]
\long\def\eql@teststaropt@loose#1#2#3{%
  \eql@ifstar@loose{\eql@testopt@loose{#1}{#3}}{\eql@testopt@loose{#2}{#3}}}
\long\def\eql@teststaropt@tight#1#2#3{%
  \eql@ifstar@tight{\eql@testopt@tight{#1}{#3}}{\eql@testopt@tight{#2}{#3}}}
\long\def\eql@teststaroropt@loose#1#2#3{%
  \eql@ifstar@loose{#1}{\eql@testopt@loose{#2}{#3}}}
\long\def\eql@teststaroropt@tight#1#2#3{%
  \eql@ifstar@tight{#1}{\eql@testopt@tight{#2}{#3}}}
\long\def\eql@gobbleopt[#1]{}
\long\def\eql@gobbleoptone[#1]#2{}
```

`\eql@spbgroup`
`\eql@spegroup`
`\eql@srbgroup`
`\eql@sregroup`

The second challenge is addressed by enclosing the look-ahead in spurious groups[3] which protect against triggering '`&`'. The macros `\eql@spbgroup` and `\eql@spegroup` open and

[3] See https://www.latex-project.org/cgi-bin/ltxbugs2html?pr=latex/3040,

close a spurious group. For some reason, the look-ahead mechanism requires further protections by inserting `\relax` at the beginning and by resetting `\@let@token` at the end. These adjustments are included in the macros `\eql@srbgroup` and `\ers@spegroup`:

```
\def\eql@spbgroup{\iffalse{\fi\ifnum0=`}\fi}
\def\eql@spegroup{\ifnum0=`{\fi\iffalse}\fi}
\def\eql@srbgroup{\relax\iffalse{\fi\ifnum0=`}\fi}
\def\eql@sregroup{\let\@let@token\relax\ifnum0=`{\fi\iffalse}\fi}
```

`\eql@ampprotect` `\eql@ampprotecttwo`

The macros `\eql@ampprotect` and `\eql@ampprotecttwo` inject the opening and closing of spurious groups into the look-ahead mechanism:

```
\long\def\eql@ampprotect#1#2{\eql@srbgroup#1{\eql@sregroup#2}}
\long\def\eql@ampprotecttwo#1#2#3{%
  \eql@srbgroup#1{\eql@sregroup#2}{\eql@sregroup#3}}
```

`...@ampsafe`

We introduce a collection of ‘`&`’-safe look-ahead macros:

```
\def\eql@ifnextchar@loose@ampsafe#1{%
  \eql@ampprotecttwo{\eql@ifnextchar@loose#1}}
\def\eql@ifnextchar@tight@ampsafe#1{%
  \eql@ampprotecttwo{\eql@ifnextchar@tight#1}}
\def\eql@ifstar@loose@ampsafe{\eql@ampprotecttwo\eql@ifstar@loose}
\def\eql@ifstar@tight@ampsafe{\eql@ampprotecttwo\eql@ifstar@tight}
\def\eql@testopt@loose@ampsafe{\eql@ampprotect\eql@testopt@loose}
\def\eql@testopt@tight@ampsafe{\eql@ampprotect\eql@testopt@tight}
\def\eql@teststaropt@loose@ampsafe{\eql@ampprotecttwo\eql@teststaropt@loose}
\long\def\eql@teststaropt@tight@ampsafe{%
  \eql@ampprotecttwo\eql@teststaropt@tight}
```

`\eql@amproof` `\eql@amprevert`

We may want to replace LaTeX’s definitions `\@ifnextchar`, `\@ifstar` and `\@testopt` to respect ‘`&`’ characters within aligned equations. This might make unrelated definitions with optional arguments and starred variants more robust in this context. The macro `\eql@amproof` overwrites the original definitions, and `\eql@amprevert` reverts the changes:

```
\let\eql@ifnextchar@org\@ifnextchar
\let\eql@ifstar@org\@ifstar
\let\eql@testopt@org\@testopt
\def\eql@amprevert{%
  \let\@ifnextchar\eql@ifnextchar@org
  \let\@testopt\eql@testopt@org
  \let\@ifstar\eql@ifstar@org
}
\def\eql@ampproof{%
  \let\@ifnextchar\eql@ifnextchar@loose@ampsafe
  \let\@testopt\eql@testopt@loose@ampsafe
  \let\@ifstar\eql@ifstar@loose@ampsafe
}
```

B.5 Error Messages

`\eql@error` `\eql@warning`

Main error and warning message function for the package:

```
\def\eql@error#1{\PackageError{eqnlines}{#1}{}}
\def\eql@warning{\PackageWarning{eqnlines}}
```

`https://www.latex-project.org/cgi-bin/ltxbugs2html?pr=amslatex/1834` and `https://tex.stackexchange.com/questions/9897/showcase-of-brace-tricks-egroup-iffalse-fi-etc`.

eql@error@nomathmode
\eql@error@mathmode

Error messages concerning math mode:

```
\def\eql@error@nomathmode#1{\eql@error{#1 allowed only in math mode}}
\def\eql@error@mathmode#1{\eql@error{#1 allowed only in paragraph mode}}
```

ql@warn@label@unused
@warn@label@multiple
\eql@warn@tag@unused
ql@warn@tag@multiple
arn@labelname@unused
n@labelname@multiple

Warning messages concerning unused and multiply declared labels and tags:

```
\def\eql@warn@label@unused{\eql@warning{Unused equation \string\label:
      label ‘\eql@nextlabel’ will be lost}}
\def\eql@warn@label@multiple#1#2{\eql@warning{%
      Multiple equation \string\label’s:
      previous label ’#1’ will be lost in favor of ’#2’}}
\def\eql@warn@labelname@unused{\eql@warning{Unused equation label name:
      name declaration will be lost}}
\def\eql@warn@labelname@multiple{\eql@warning{Multiple equation label names:
      previous name declaration will be lost}}
\def\eql@warn@tag@unused{\eql@warning{Unused equation \string\tag:
      tag declaration will be lost}}
\def\eql@warn@tag@multiple{\eql@warning{Multiple equation \string\tag’s:
      previous tag declaration will be lost}}
```

B.6 amsmath Integration

\eql@amsmath@after
\eql@amsmath@before
eql@amsmath@undefine
\eql@amsmath@let

We need to overwrite certain macros from amsmath. The method `\eql@amsmath@after` executes argument `#1` after loading amsmath is loaded. It also runs the code if amsmath has already been loaded. Furthermore, loading amsmath requires certain macros to be undefined. To this end `\eql@amsmath@before` will execute argument `#1` before any future loading of amsmath. `\eql@amsmath@undefine` undefines a macro in this way and `\eql@amsmath@let` overwrites a macro of `\amsmath/`:

```
\def\eql@amsmath@after#1{\AddToHook{package/amsmath/after}{#1}}
\def\eql@amsmath@before#1{%
  \@ifpackageloaded{amsmath}{}{\AddToHook{package/amsmath/before}{#1}}}
\def\eql@amsmath@undefine#1{\eql@amsmath@before{\let#1\@undefined}}
\def\eql@amsmath@let#1#2{\eql@amsmath@undefine#1\let#1#2}
```

TODO: temporary fix for development stages

```
\@ifpackageloaded{amsmath}{}{
  \DeclareHookRule{package/amsmath/after}
    {eqnlines}{after}{latex-lab-testphase-math}}
```

B.7 PDF Tagging Support

\eql@tagging@...

Proper pdf tagging[4] support requires a LaTeX (development) version at least of 2025. For the time being, we define an abstraction layer so that the package will collaborate with LaTeX versions around 2020: **TODO:** adjust to further developments

```
\let\eql@tagging@on\eql@false
\IfFormatAtLeastTF{2025-06-01}{%
  \csname tag_if_active:T\endcsname{\let\eql@tagging@on\eql@true}}{}
\ifdefined\eql@tagging@on
  \def\eql@tagging@mathsave{%
    \UseTaggingSocket{math/luamml/save/nNn}{{}\displaystyle{mtd}}}
  \def\eql@tagging@mathaddlast{%
    \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}}
```

[4]see `https://latex3.github.io/tagging-project/`

```
  \def\eql@tagging@tagbegin{%
    \UseTaggingSocket{math/display/tag/begin}}
  \def\eql@tagging@tagend{%
    \UseTaggingSocket{math/display/tag/end}}
  \def\eql@tagging@tagsave{%
    \UseTaggingSocket{math/luamml/mtable/tag/save}}
  \def\eql@tagging@tagaddbox{%
    \setbox\z@\copy\eql@tagbox@%
    \UseTaggingSocket{math/luamml/mtable/tag/set}}
  \def\eql@tagging@tablesaveinner{%
    \UseExpandableTaggingSocket{math/luamml/mtable/innertable/save}}
  \def\eql@tagging@tableaddinner{%
    \UseTaggingSocket{math/luamml/mtable/innertable/finalize}}
  \def\eql@tagging@tablesavelines{%
    \UseExpandableTaggingSocket{math/luamml/mtable/finalize}{gather}}
  \def\eql@tagging@tablesavealign{%
    \UseExpandableTaggingSocket{math/luamml/mtable/finalize}{align}}
  \def\eql@tagging@alignleft{%
    \UseTaggingSocket{math/luamml/mtable/aligncol}{left}}
  \def\eql@tagging@aligncenter{%
    \UseTaggingSocket{math/luamml/mtable/aligncol}{center}}
  \def\eql@tagging@alignright{%
    \UseTaggingSocket{math/luamml/mtable/aligncol}{right}}
```

We need to get hold of the equation body in all cases so that we can feed it into the tagging mechanism:

```
  \let\eql@single@doscan\eql@true
  \let\eql@scan@body\eql@scan@body@rescan
```

`\eql@tagging@start`
`\eql@tagging@end`

We need to activate tagging for display equations for environments and for enclosures `\[...\]` and `\<...\>`. The tagging interface registration macro `\RegisterMathEnvironment` will work only partially for our cases, hence we replicate code from `\math_register_halign_env:nn`. Make sure collection is not yet active (`\l__math_collected_bool`). Then feed collected environment name, options and body into `\__math_process:nn`. Indicate the start of a display equation:

```
  \def\eql@tagging@start{%
    \csname bool_if:NF\expandafter\endcsname
      \csname l__math_collected_bool\endcsname{%
      \toks@\expandafter{\eql@tagging@opt}%
      \edef\eql@tmp{{\@currenvir}{[\the\toks@] \the\eql@scan@reg@}}%
      \csname __math_process:nn\expandafter\endcsname\eql@tmp
      \@kernel@math@registered@begin
      \csname bool_set_true:N\expandafter\endcsname
        \csname l__math_collected_bool\endcsname
    }%
  }
  \def\eql@tagging@end{}
  \def\eql@tagging@register@env{\csname math_register_env:n\endcsname}
\else
  \def\eql@tagging@mathsave{}
  \def\eql@tagging@mathaddlast{}
  \def\eql@tagging@tagbegin{}
  \def\eql@tagging@tagend{}
  \def\eql@tagging@tagsave{}
  \def\eql@tagging@tagaddbox{}
  \def\eql@tagging@tablesaveinner{}
  \def\eql@tagging@tableaddinner{}
  \def\eql@tagging@tablesavelines{}
  \def\eql@tagging@tablesavealign{}
  \def\eql@tagging@alignleft{}
  \def\eql@tagging@aligncenter{}
  \def\eql@tagging@alignright{}
  \def\eql@tagging@start{}
  \def\eql@tagging@end{}
  \def\eql@tagging@register@env{\@gobble}
\fi
```

C Parameters and Registers

In the following, we collect parameter and register definitions.

C.1 Parameters

TODO: describe

TODO: maybe sort parameters into sections **TODO:** or sort parameters in sections here

`\eql@tagsleft` (*bool*) The boolean parameter `\eql@tagsleft` specifies whether the tags are placed at the left or right margin:

```
\let\eql@tagsleft\eql@false
```

`\eql@layoutleft` (*bool*) The boolean parameter `\eql@layoutleft` specifies whether to use left or central alignment layout:

```
\let\eql@layoutleft\eql@false
```

`eql@layoutleftmargin`
`@layoutleftmarginmin`
`@layoutleftmarginmax`
The default width of the left margin in left alignment layout is specified by `\eql@layoutleftmargin`. It may be pushed down to `\eql@layoutleftmarginmin` and up to `\eql@layoutleftmarginmax`:

```
\def\eql@layoutleftmargin{\leftmargini}
\def\eql@layoutleftmarginmax{.5\maxdimen}
\def\eql@layoutleftmarginmin{\z@}
```

`ql@tagmargin@` (*dimen*)
`argin@ratio@` (*dimen*)
`\eql@tagmargin@val`
`@tagmargin@threshold`
The intended margin width for tags in centeral alignment layout is stored in `\eql@tagmargin@` which is sourced by `\eql@tagmargin@val`. An undefined `\eql@tagmargin@val` will compute the margin width as the maximum width of tags (without separation). `\eql@tagmargin@ratio@` describes the maximum ratio of lines with tags to total number of lines for which `\eql@tagmargin@` is set to zero: **TODO:** threshold

```
\newdimen\eql@tagmargin@
\let\eql@tagmargin@val\@undefined
\newdimen\eql@tagmargin@ratio@
\eql@tagmargin@ratio@\p@
\def\eql@tagmargin@threshold{0.5}
```

`\eql@indent@` (*dimen*) The currently selected indentation width is speficied by `\eql@indent@`. This dimension register is set to the macro `\eql@indent@val` when entering the equation environments:

```
\newdimen\eql@indent@
\def\eql@indent@val{2em}
```

@paddingleft@ (*dimen*)
addingright@ (*dimen*)

The padding of an equation (column) is specified by `\eql@paddingleft@` and `\eql@paddingright@`. These dimension registers are set to the macros `\eql@paddingleft@val` and `\eql@paddingright@val`, respectively, when entering the equation environments:

```
\newdimen\eql@paddingleft@
\newdimen\eql@paddingright@
\let\eql@paddingleft@val\@undefined
\let\eql@paddingright@val\@undefined
```

ql@display@linewidth
l@display@marginleft
@display@marginright

TODO: describe

```
\let\eql@display@linewidth\@undefined
\let\eql@display@marginleft\@undefined
\let\eql@display@marginright\@undefined
```

\eql@box@colsep

The macro `\eql@box@colsep` specifies the intercolumn separation for equation boxes:

```
\def\eql@box@colsep{2em}
```

\eql@spread@val

The extra spread of equation lines is specified by `\eql@spread@val`:

```
\def\eql@spread@val{\jot}
\newdimen\eql@spread@
```

\eql@tagfuzz@ (*dimen*)

The value `\eql@tagfuzz@` specifies the margin of error for comparing whether a tag fits a given equation line. We should not expect rounding errors in the fixed point arithmetic of TEX, nevertheless: **TODO:** probably do not need this due to fixed point arithmetic.

```
\newdimen\eql@tagfuzz@
\eql@tagfuzz@16sp\relax
```

\eql@display@height
\eql@display@depth

An equation will appear to the surrounding text with a fixed apparent height and depth specified by `\eql@display@height` and `\eql@display@depth`, respectively:

```
\def\eql@display@height\@undefined
\def\eql@display@depth\@undefined
```

\eql@skip@mode@short

The setting `\eql@skip@mode@short` specifies when a reduced amount of glue should be used around equations in case the text line above the equation fits in the space that is left available in the first equation line. Value `0` turns this feature off, value `1` reduces the glue above the equation, value `2` furthermore reduces the glue below a single equation line and value `3` also reduces the glue below multi-line equations:

```
\def\eql@skip@mode@short{2}

\def\eql@skip@mode@cont@above{2}
\def\eql@skip@mode@cont@below{0}

\def\eql@skip@mode@par@above{3}
\def\eql@skip@mode@par@below{0}

\def\eql@skip@mode@top@above{4}
\def\eql@skip@mode@top@below{0}

\newcount\eql@skip@mode@leave@
\let\eql@skip@force@leave\@undefined
```

`eql@skip@force@above` `eql@skip@force@below` `mode@above@` (*counter*) `mode@below@` (*counter*)

0: short, 1: long, 2: cont, 3: par, 4: top, 5: no, 6: med, 7: custom

```
\newcount\eql@skip@mode@above@
\newcount\eql@skip@mode@below@
\let\eql@skip@force@above\@undefined
\let\eql@skip@force@below\@undefined
\let\eql@skip@custom@above\@undefined
\let\eql@skip@custom@below\@undefined
```

`\eql@skip@cont@above` The glue when an equation is at the top of a horizontal list is specified by `\eql@skip@cont@above`:

`\eql@skip@top@above` `\eql@skip@top@below` The glue when an equation is at the top of a vertical list is specified by `\eql@skip@top@above` and `\eql@skip@top@below`:

`\eql@skip@par@above` The glue when an equation starts a paragraph is specified by `\eql@skip@par@above`:

`\eql@skip@med@above` `\eql@skip@med@below` The surrounding glue for an equation with reduced spacing is given by `\eql@skip@med@above` and `\eql@skip@med@below`:

```
\def\eql@skip@long@above{\abovedisplayskip}
\def\eql@skip@long@below{\belowdisplayskip}
\def\eql@skip@short@above{\abovedisplayshortskip}
\def\eql@skip@short@below{\belowdisplayshortskip}
\def\eql@skip@cont@above{\eql@skip@short@above}
\def\eql@skip@cont@below{\eql@skip@short@below}
% \TODO parabove plus parskip?
\def\eql@skip@par@above{\eql@skip@long@above}
\def\eql@skip@par@below{\eql@skip@long@below}
\def\eql@skip@top@above{\eql@skip@long@above}
\def\eql@skip@top@below{\eql@skip@long@below}
\def\eql@skip@med@above{\abovedisplayskip/2}
\def\eql@skip@med@below{\belowdisplayskip/2}
\def\eql@skip@tag@above{\z@skip}
\def\eql@skip@tag@below{\z@skip}
\def\eql@skip@partag@above{\z@skip}
\def\eql@skip@partag@below{\z@skip}
\def\eql@skip@medtag@above{\z@skip}
\def\eql@skip@medtag@below{\z@skip}
```

`l@colsepmin@` (*dimen*) `\eql@colsepmin@val` `\eql@colsepmax@val` The minimum intercolumn separatation is specified by `\eql@colsepmin@`. This dimension register is set to `\eql@colsepmin@val` when entering the equation environments to allow font-dependent values. Furthermore, `\eql@colsepmax@val` specifies the maximum intercolumn separation:

```
\newdimen\eql@colsepmin@
\def\eql@colsepmin@val{1em}
\def\eql@colsepmax@val{.5\maxdimen}
```

`@tagwidthmin@` (*dimen*) The minimum tag width is specified by `\eql@tagwidthmin@`:

```
\newdimen\eql@tagwidthmin@
\eql@tagwidthmin@\z@
```

`l@tagsepmin@` (*dimen*) The minimum separation between an equation and its tag is given by `\eql@tagsepmin@`. TeX's built-in value is half a quad[5] in font number 2. As the tag is processed in text mode, we use `0.5em` instead.

[5]another half of a quad is left empty at the other end of the line.

```
\newdimen\eql@tagsepmin@
\def\eql@tagsepmin@val{.5\fontdimen6\textfont\tw@}
```

ql@equations@sqr@opt
ql@equations@ang@opt
\eql@box@ang@opt

Store the default arguments for `\[...\]` and `\<...\>`, respectively:

```
\def\eql@equations@sqr@opt{equation,nonumber}
\def\eql@equations@ang@opt{align,nonumber}
\def\eql@box@ang@opt{align}
```

Multi-Line Align Mode.

```
\let\eql@columns@fulllength\eql@false
```

C.2 Registers

TODO: describe

General.

\eql@cellbox@ (*box*)
\eql@tagbox@ (*box*)
ql@cellwidth@ (*dimen*)
ql@prevwidth@ (*dimen*)
eql@tagwidth@ (*dimen*)
ql@prevdepth@ (*dimen*)
l@prevgraf@ (*counter*)

The box `\eql@cellbox@` holds the present alignment component and `\eql@tagbox@` the tag for the present line. The corresponding dimensions `\eql@cellwidth@` and `\eql@tagwidth@` hold their widths. `\eql@prevwidth@` holds the width of the previous alignment component: **TODO:** adjust

```
\newbox\eql@cellbox@
\newbox\eql@tagbox@
\newdimen\eql@cellwidth@
\newdimen\eql@prevwidth@
\newdimen\eql@tagwidth@
\newdimen\eql@prevdepth@
\newcount\eql@prevgraf@
```

l@totalwidth@ (*dimen*)
agwidth@max@ (*dimen*)

```
\newdimen\eql@totalwidth@
\newdimen\eql@tagwidth@max@
```

line@height@ (*dimen*)
l@line@depth@ (*dimen*)

The dimension registers `\eql@line@height@` and `\eql@line@depth@` keep track of the height and depth of the present line in an alignment:

```
\newdimen\eql@line@height@
\newdimen\eql@line@depth@
```

l@line@width@ (*dimen*)
l@line@avail@ (*dimen*)
eql@line@pos@ (*dimen*)
e@widthsep@ (*counter*)
e@availsep@ (*counter*)
ine@possep@ (*counter*)
line@offset@ (*dimen*)

```
\newdimen\eql@line@width@
\newdimen\eql@line@avail@
\newdimen\eql@line@pos@
\newcount\eql@line@availsep@
\newcount\eql@line@widthsep@
\newcount\eql@line@possep@
\newdimen\eql@line@offset@
```

Rows and Columns.

`\eql@row@` (*counter*)
`.@totalrows@` (*counter*)
`ql@tagrows@` (*counter*)

`\eql@row@` counts the present row (1-based) and `\eql@totalrows@` holds the total number of rows:

```
\newcount\eql@row@
\newcount\eql@totalrows@
\newcount\eql@tagrows@
```

`\eql@column@`
`\eql@totalcolumns@`

```
\newcount\eql@column@
\newcount\eql@totalcolumns@
```

`\eql@colsep@` (*dimen*)

The dimension of the intercolumn separation for align environments is stored in `\eql@colsep@`:

```
\newdimen\eql@colsep@
```

`umns@inter@` (*counter*)

```
\newcount\eql@columns@inter@
```

Vertical Spacing Adjustments.

`@firstavail@` (*dimen*)
`splay@firstavail@set`

The unused space on the first line of an alignment is stored in `\eql@display@firstavail@` for comparison against `\predisplaysize` and determining short skip mode of display equations. It it convenient to set it via `\eql@display@firstavail@set` provided that we are on the first line:

```
\newdimen\eql@display@firstavail@
\def\eql@display@firstavail@set#1{%
  \ifnum\eql@row@=\@ne
    \global\eql@display@firstavail@#1%
  \fi
}
```

`@firstlast@` (*counter*)

The counter stores whether the tag one first/last line is raised/lowered as `1`/`2` (or `3` for both). This implies a different vskip corresponding to the mostly empty line:

```
\newcount\eql@raisetag@firstlast@
```

Shared Registers.

`\ifmeasuring@` (*bool*)

All display environments get typeset twice – once during a “measuring” phase and then again during a “production” phase. We reuse the original amsmath definition `\ifmeasuring@` to determine which case we’re in, so we and other packages may take appropriate action. It does not hurt to define this conditional in any case. We should tell hyperref about measuring processes as we’re not amsmath and not being catered for:

```
\ifdefined\measuring@true\else
  \expandafter\newif\csname ifmeasuring@\endcsname
\fi
\AddToHook{package/hyperref/after}{
  \ifdefined\Hy@ifnotmeasuring
    \renewcommand\Hy@ifnotmeasuring[1]{\ifmeasuring@\else#1\fi}
  \fi
}
```

\if@display (*bool*) amsmath defines the conditional `\if@display` to test whether we're in a display equation including the inner math parts of equation blocks. We provide it in case amsmath is absent, and initialise it:

```
\ifdefined\@displaytrue\else
  \expandafter\newif\csname if@display\endcsname
  \everydisplay\expandafter{\the\everydisplay\@displaytrue}
\fi
```

C.3 Hooks

\eql@hook@... For what it's worth, we define a couple of entry points where one might hook into the equations typesetting framework. The LaTeX hook framewould would be more versatile, but as the purpose of these hooks is rather unclear at the moment, we make this as efficient as it could get: **TODO:** may add a few more hooks

```
\let\eql@hook@blockbefore\@empty
\let\eql@hook@blockafter\@empty
\let\eql@hook@blockin\@empty
\let\eql@hook@blockout\@empty
\let\eql@hook@linein\@empty
\let\eql@hook@lineout\@empty
\let\eql@hook@colin\@empty
\let\eql@hook@colout\@empty
\let\eql@hook@eqin\@empty
\let\eql@hook@eqout\@empty
\let\eql@hook@innerleft\@empty
\let\eql@hook@innerright\@empty
\let\eql@hook@innerlead\@empty
```

D Features

D.1 Punctuation

The equations environments supply an automatic punctuation scheme which allows to define a default punctuation at the end of each column, line and equation block.

\eql@punct@col
\eql@punct@line
\eql@punct@block

These macros store the punctuation character for columns, lines and blocks. A value `\relax` indicates that the punctuation should be handed down to the next lower level:

```
\let\eql@punct@col\@empty
\let\eql@punct@line\relax
\let\eql@punct@block\relax
```

\eql@punct@sep This macro stores the separation to be applied before the punctuation (unless it is empty):

```
\let\eql@punct@sep\relax
```

\eqnpunctcol
\eqnpunctline
\eqnpunctdefault
\eqnpunct

Set the punction for columns, lines and blocks. Note that the macro `\eqnpunct` sets the punctuation for the following equation block or for the current cell. Starred versions clear the punctuation for the respectively levels:

```
\def\eqnpunctcol{%
  \eql@ifstar@tight\eql@punct@col@setrelax\eql@punct@col@set}
\def\eql@punct@col@set#1{\def\eql@punct@col{#1}\ignorespaces}
```

```
\def\eql@punct@col@setrelax{\let\eql@punct@col\@empty\ignorespaces}
\def\eqnpunctline{%
  \eql@ifstar@tight\eql@punct@line@setrelax\eql@punct@line@set}
\def\eql@punct@line@set#1{\def\eql@punct@line{#1}\ignorespaces}
\def\eql@punct@line@setrelax{\let\eql@punct@line\relax\ignorespaces}
\def\eqnpunctdefault{%
  \eql@ifstar@tight\eql@punct@main@setrelax\eql@punct@main@set}
\def\eql@punct@main@set#1{\def\eql@punct@main{#1}\ignorespaces}
\def\eql@punct@main@setrelax{\let\eql@punct@main\relax\ignorespaces}
\def\eqnpunct{%
  \eql@ifstar@tight\eql@punct@next@setrelax\eql@punct@next@set}
\def\eql@punct@next@set#1{%
  \ifmmode
    \def\eql@punct@col{#1}%
    \def\eql@punct@line{#1}%
    \def\eql@punct@block{#1}%
  \else
    \eqnaddopt{punct={#1}}%
  \fi
  \ignorespaces}
\def\eql@punct@next@setrelax{%
  \ifmmode
    \let\eql@punct@block\relax
  \else
    \eqnaddopt{punct*}%
  \fi
  \ignorespaces}
```

\eql@punct@apply@col Output the punctuation for the present column. If non-empty, prepend some separation. Clear the punctuation so that no further column punctuation is output within the current group:

```
\def\eql@punct@apply@col{%
  \ifx\eql@punct@col\@empty\else
    \eql@punct@sep
    \eql@punct@col
    \let\eql@punct@col\@empty
  \fi
}
```

Output the punctuation currently set for lines unless disabled. Alike `\eql@punct@apply@col` prevent further output of punctuations for lines and columns within the current group:

eql@punct@apply@line

```
\def\eql@punct@apply@line{%
  \ifx\eql@punct@line\relax
% \TODO hand down immediately?
  \else
    \ifx\eql@punct@line\@empty\else
      \eql@punct@sep
      \eql@punct@line
    \fi
    \let\eql@punct@line\relax
    \let\eql@punct@col\@empty
  \fi
}
```

ql@punct@apply@block
\eqnpunctapply

Outputs the punctuation for the current equation block unless disabled in analogy to `\eql@punct@apply@line`:

```
\def\eql@punct@apply@block{%
  \ifx\eql@punct@block\relax
% \TODO hand down immediately?
  \else
    \ifx\eql@punct@block\@empty\else
      \eql@punct@sep
      \eql@punct@block
    \fi
    \let\eql@punct@block\relax
    \let\eql@punct@line\relax
    \let\eql@punct@col\@empty
  \fi
}
\let\eqnpunctapply\eql@punct@apply@block
```

D.2 Math Classes at Alignment

The following describes the adjustment of math classes surrounding the alignment marker.

lass@innerright@sel@

Select between `\eql@class@innerlead` and `\eql@class@innerright` depending on whether the left part of the aligned column is empty:

```
\def\eql@class@innerright@sel@{%
  \ifdim\eql@prevwidth@=\z@
    \eql@class@innerlead
  \else
    \eql@class@innerright
  \fi
}
```

@class@innerleft@set
class@innerright@set
@class@innerlead@set

Set the left, right and leading math classes. Setting the right math class disables the leading math class, so the leading math class must be specified after the right one:

```
\def\eql@class@innerleft@set#1{%
  \def\eql@class@innerleft{#1}%
}
\def\eql@class@innerright@set#1{%
  \def\eql@class@innerright{#1}%
  \let\eql@class@innerright@sel\eql@class@innerright
}
\def\eql@class@innerlead@set#1{%
  \def\eql@class@innerlead{#1}%
  \let\eql@class@innerright@sel\eql@class@innerright@sel@
}
```

\eql@class@ampeq
\eql@class@eqamp

We define two standard combinations of math classes intended to be used with '`&=`' (`ampeq`) or '`=&`' (`eqamp`). The default setting is '`&=`' (`ampeq`):

```
\def\eql@class@ampeq{%
  \eql@class@innerleft@set{}%
  \eql@class@innerright@set{{}}%
}
\def\eql@class@eqamp{%
  \eql@class@innerleft@set{\mathrel{}}%
  \eql@class@innerright@set{\mathrel{}}%
  \eql@class@innerlead@set{{}}%
}
\eql@class@ampeq
```

D.3 Framed Cells

TODO: describe **TODO:** warn if issued in even cells

```
\let\eql@frame@cmd\@undefined
\newdimen\eql@frame@margin@
\def\eql@frame@set[#1]{\global\def\eql@frame@cmd{#1}}
\def\eql@frame@reset{\global\let\eql@frame@cmd\@undefined}
\protected\def\framecell{\eql@testopt@tight@ampsafe\eql@frame@set\fbox}
\def\eql@frame@measure{%
  \setbox\z@\hbox{\eql@frame@cmd{}}%
  \eql@frame@margin@.5\wd\z@
}
\def\eql@frame@print{%
  \setbox\eql@cellbox@\hbox{%
    \eql@frame@cmd{\unhbox\eql@cellbox@}%
  }%
  \eql@frame@reset
}
\def\eql@frame@adjust{%
  \setbox\eql@cellbox@\hbox{%
    \eql@frame@cmd{%
      \unhbox\eql@cellbox@
      \unkern
      \unskip
    }%
    \hfil
    \kern\z@
  }%
  \eql@frame@reset
}
```

D.4 Alternative Content Description

TODO: describe **TODO:** would be nice to provide as environments as well **TODO:** implement for pdf tagging

```
\DeclareRobustCommand{\eqnalt}[2][]{}
```

E Equation Numbering

TODO: describe

E.1 Supporting Definitions

Parameters.

```
\let\eql@numbering@autolabel\eql@false
\let\eql@numbering@autotag\eql@true
\let\eql@numbering@warn\eql@true
```

```
\def\eql@labelname@generic{[equation]}
```

Registers.

```
\let\eql@numbering@mode\@undefined
\let\eql@numbering@measure@call\@empty
```

`ing@target@` (*counter*)

```
\let\eql@numbering@active\eql@true
\newcount\eql@numbering@target@
```

```
\let\eql@nextlabel\@undefined
\let\eql@nextlabelname\@undefined
\let\eql@blocklabel\@undefined
\let\eql@blocklabelname\@undefined
```

`\eql@nexttag`

```
\let\eql@nexttag\@undefined
\let\eql@blocktag\@undefined
```

`setag@amount@` (*dimen*)

```
\newdimen\eql@raisetag@amount@
```

`g@refcount@` (*counter*)

```
\newcount\eql@numbering@refcount@
\eql@numbering@refcount@\z@
\def\eql@numbering@ref{equation.eql-\the\eql@numbering@refcount@}
\def\eql@numbering@refstep{\global\advance\eql@numbering@refcount@\@ne}
```

E.2 Schemes

TODO: describe

Table.

```
\def\eql@numbering@tab@first{first}
\def\eql@numbering@tab@last{last}
\def\eql@numbering@tab@middle{middle}
\def\eql@numbering@tab@here{here}
\def\eql@numbering@tab@best{best}
\def\eql@numbering@tab@in{in}
\def\eql@numbering@tab@out{out}
\def\eql@numbering@tab@sub{sub}
\def\eql@numbering@tab@all{all}
\def\eql@numbering@tab@none{none}
```

TODO: describe

```
\let\eql@numbering@tab@f\eql@numbering@tab@first
\let\eql@numbering@tab@l\eql@numbering@tab@last
\let\eql@numbering@tab@m\eql@numbering@tab@middle
\let\eql@numbering@tab@mid\eql@numbering@tab@middle
\let\eql@numbering@tab@o\eql@numbering@tab@out
```

```
\let\eql@numbering@tab@outside\eql@numbering@tab@out
\let\eql@numbering@tab@i\eql@numbering@tab@in
\let\eql@numbering@tab@inside\eql@numbering@tab@in
\let\eql@numbering@tab@within\eql@numbering@tab@in
\let\eql@numbering@tab@h\eql@numbering@tab@here
\let\eql@numbering@tab@optimal\eql@numbering@tab@best
\let\eql@numbering@tab@s\eql@numbering@tab@sub
\let\eql@numbering@tab@subeq\eql@numbering@tab@sub
\let\eql@numbering@tab@subequation\eql@numbering@tab@sub
\let\eql@numbering@tab@subequations\eql@numbering@tab@sub
\let\eql@numbering@tab@a\eql@numbering@tab@all
\let\eql@numbering@tab@n\eql@numbering@tab@none
\expandafter\let\csname eql@numbering@tab@!\endcsname\eql@numbering@tab@all
\expandafter\let\csname eql@numbering@tab@*\endcsname\eql@numbering@tab@none
\expandafter\let\csname eql@numbering@tab@1\endcsname\eql@numbering@tab@first
\expandafter\let\csname eql@numbering@tab@+\endcsname\eql@numbering@tab@best
\let\eql@numbering@mode\eql@numbering@tab@all
```

Implementations.

```
\def\eql@numbering@mode@all{%
  \eql@numbering@target@\m@ne}
\def\eql@numbering@mode@first{%
  \eql@numbering@target@\@ne}
\def\eql@numbering@mode@last{%
  \eql@numbering@target@\@MM}
\def\eql@numbering@mode@here{%
  \eql@numbering@target@\z@}
```

TODO: describe

```
\def\eql@numbering@mode@in{%
  \ifdefined\eql@tagsleft
    \eql@numbering@mode@last
  \else
    \eql@numbering@mode@first
  \fi
}
```

TODO: describe

```
\def\eql@numbering@mode@out{%
  \ifdefined\eql@tagsleft
    \eql@numbering@mode@first
  \else
    \eql@numbering@mode@last
  \fi
}
```

TODO: describe

```
\def\eql@numbering@mode@middle{%
  \eql@numbering@target@\z@
  \let\eql@numbering@measure@call\eql@numbering@measure@middle}
\def\eql@numbering@measure@middle{%
  \ifnum\eql@numbering@target@=\z@
    \eql@numbering@target@\eql@totalrows@
    \advance\eql@numbering@target@\@ne
    \divide\eql@numbering@target@\tw@
  \fi
}
```

TODO: describe

```
\def\eql@numbering@mode@best{%
  \eql@numbering@target@\z@
  \let\eql@numbering@measure@call\eql@numbering@measure@best
}
\def\eql@numbering@measure@best{%
  \ifnum\eql@numbering@target@=\z@
    \let\eql@numbering@best@use\eql@true
    \eql@numbering@mode@out
  \fi
}
```

TODO: describe

```
\def\eql@numbering@mode@sub{%
  \eql@numbering@target@\m@ne
  \ifdefined\eql@subequations@active\else
    \let\eql@numbering@measure@call\eql@numbering@subeq@test
    \let\eql@numbering@subeq@use\eql@true
  \fi
}
```

Selection.

```
\def\eql@numbering@set#1{%
  \ifcsname eql@numbering@tab@#1\endcsname
    \expandafter\let\expandafter\eql@numbering@mode
      \csname eql@numbering@tab@#1\endcsname
    \ifx\eql@numbering@mode\eql@numbering@tab@none
      \let\eql@numbering@mode\eql@numbering@tab@all
      \let\eql@numbering@active\eql@false
    \fi
  \else
    \eql@error{numbering mode ‘#1’ unknown: setting to ‘all’}%
    \let\eql@numbering@mode\eql@numbering@tab@all
  \fi
}
```

TODO: describe

```
\def\eql@numbering@mode@eval{%
  \let\eql@numbering@measure@call\@empty
  \let\eql@numbering@subeq@use\eql@false
  \csname eql@numbering@mode@\eql@numbering@mode\endcsname
  \ifdefined\eql@numbering@active
    \let\eql@numbering@eqnswinit\@eqnswtrue
  \else
    \let\eql@numbering@eqnswinit\@eqnswfalse
  \fi
  \let\eql@numbering@active\eql@false
}
```

E.3 Interface

Activation. **TODO:** note `\nonumber` already defined, modifications by amsmath

```
\eql@amsmath@after{
  \let\eql@print@eqnum@default\print@eqnum
  \let\eql@incr@eqnum@default\incr@eqnum
}
```

TODO: describe

```
\protected\def\donumber{%
  \if@eqnsw\else
    \global\@eqnswtrue
    \ifx\print@eqn\@empty
      \global\let\print@eqn\eql@print@eqnum@default
    \fi
    \ifx\incr@eqn\@empty
      \global\let\incr@eqn\eql@incr@eqnum@default
    \fi
  \fi
}
```

TODO: reconsider operation

`\numberhere`

```
\protected\def\eql@numberhere{%
  \global\eql@numbering@target@\eql@row@
}
```

TODO: describe

`\numbernext`

```
\protected\def\eql@numbernext{%
  \ifnum\eql@numbering@target@=\eql@row@
    \global\advance\eql@numbering@target@\@ne
  \fi
}
```

Activation Trigger.

```
\def\eql@numbering@autoenable{%
  \global\@eqnswtrue
  \ifx\eql@numbering@mode\eql@numbering@tab@here
    \ifnum\eql@numbering@target@=\z@
      \numberhere
    \fi
  \fi
}
```

Labels. **TODO:** describe

`\eql@label@org`

```
\let\eql@label@org\label
```

TODO: describe

```
\def\eql@label@gobble{\eql@ampprotect\eql@testopt@tight\eql@gobbleoptone{}}
```

TODO: describe

```
\protected\def\eql@label{%
  \ifdefined\eql@numbering@autolabel
    \eql@numbering@autoenable
  \fi
  \eql@ampprotecttwo{\eql@ifnextchar@tight[}\eql@label@name\eql@label@plain%]
}
```

TODO: describe

```
\def\eql@label@plain#1{%
  \ifdefined\eql@numbering@warn
    \ifdefined\eql@nextlabel
      \eql@warn@label@multiple\eql@nextlabel
    \fi
  \fi
  \global\edef\eql@nextlabel{#1}%
}
```

TODO: describe

```
\def\eql@label@name[#1]#2{%
  \ifdefined\eql@numbering@warn
    \ifdefined\eql@nextlabel
      \eql@warn@label@multiple\eql@nextlabel{#2}%
    \fi
    \ifdefined\eql@nextlabelname
      \eql@warn@labelname@multiple
    \fi
  \fi
  \protected@edef\eql@nextlabelname{#1}%
  \global\let\eql@nextlabelname\eql@nextlabelname
  \global\edef\eql@nextlabel{#2}%
}
```

TODO: describe

```
\def\eql@blocklabel@set#1{%
  \ifdefined\eql@blocklabel
    \eql@warn@label@multiple\eql@blocklabel{#1}%
  \fi
  \edef\eql@blocklabel{#1}%
}
```

TODO: describe

```
\def\eql@blocklabelname@set#1{%
  \ifdefined\eql@blocklabelname
    \eql@warn@labelname@multiple
  \fi
  \protected@edef\eql@blocklabelname{#1}%
}
```

Tags. **TODO:** describe

\eql@tag@default

```
\protected\def\eql@tag@default{%
  \eql@error{\string\tag\space not allowed here}{}\eql@tag@gobble}
\let\tag\eql@tag@default
```

\eql@tag@gobble

```
\def\eql@tag@gobble{%
  \eql@ampprotecttwo\eql@teststaropt@tight\eql@gobbleoptone\eql@gobbleoptone{}}
```

TODO: describe

```
\protected\def\eql@tag{%
  \ifdefined\eql@numbering@autotag
    \eql@numbering@autoenable
  \fi
  \ifdefined\eql@numbering@warn
    \ifdefined\eql@nexttag
      \eql@warn@tag@multiple
    \fi
  \fi
  \eql@ampprotecttwo\eql@teststaropt@tight
    {\eql@nexttag@set\z@}{\eql@nexttag@set\@ne}\eql@tmp
}
```

\eql@nexttag@set

```
\def\eql@nexttag@set#1[#2]#3{%
  \protected@edef\eql@tmp{#3}%
  \protected@edef\eql@nexttag{{#1}{#2}{\eql@tmp}}%
  \global\let\eql@nexttag\eql@nexttag
}
```

\eql@tag@eval

```
\def\eql@tag@eval#1#2#3{%
  \def\eql@tag@label{#2}%
  \def\eql@tag@text{#3}%
  \ifnum#1=\@ne
    \let\eql@tag@tool\eql@tag@form
    \protected@edef\eql@tag@label{\p@equation\eql@tag@label}%
  \else
    \let\eql@tag@tool\@firstofone
  \fi
}
```

TODO: describe

```
\def\eql@blocktag@set#1#2{%
  \ifdefined\eql@blocktag
    \eql@warn@tag@multiple
  \fi
  \protected@edef\eql@blocktag{{#1}{#2}{#2}}%
}
```

Raise Tags.

\raisetag

```
\def\eql@raisetag@default{%
  \eql@warning{\string\raisetag\space not allowed here}
  \@gobble
}
```

TODO: describe

```
\eql@amsmath@let\raisetag\eql@raisetag@default
```

TODO: maybe introduce a star form to enforce raise?

```
\def\eql@raisetag#1{\global\eql@raisetag@amount@\glueexpr#1\relax}
```

```
\let\eql@raisetag@gobble\@gobble
```

E.4 Integration

TODO: describe

Single Line. **TODO:** describe

```
\def\eql@numbering@single@init{%
  \let\eql@numbering@warn\eql@true
  \let\label\eql@label
  \let\tag\eql@tag
  \let\raisetag\eql@raisetag
  \let\numberhere\donumber
  \let\numbernext\nonumber
  \eql@numbering@target@\m@ne
  \let\eql@nextlabel\eql@blocklabel
  \let\eql@nextlabelname\eql@blocklabelname
  \let\eql@nexttag\eql@blocktag
  \eql@numbering@eqnswinit
  \ifdefined\eql@numbering@autolabel
    \ifdefined\eql@nextlabel
      \@eqnswtrue
    \fi
  \fi
  \ifdefined\eql@numbering@autotag
    \ifdefined\eql@nexttag
      \@eqnswtrue
    \fi
  \fi
  \global\eql@raisetag@amount@\z@
}
```

Multi-Line Measuring Pass. **TODO:** describe

```
\def\eql@numbering@measure@init{%
  \let\eql@numbering@warn\eql@true
  \let\label\eql@label
  \let\tag\eql@tag
  \let\raisetag\eql@raisetag
  \ifnum\eql@numbering@target@<\z@
    \let\numberhere\donumber
    \let\numbernext\nonumber
  \else
    \let\numberhere\eql@numberhere
    \let\numbernext\eql@numbernext
    \eql@numbering@eqnswinit
    \ifdefined\eql@numbering@autolabel
      \ifdefined\eql@nextlabel
        \@eqnswtrue
      \fi
    \fi
  \fi
  \let\eql@nextlabel\eql@blocklabel
  \let\eql@nextlabelname\eql@blocklabelname
  \let\eql@nexttag\eql@blocktag
}
```

TODO: might select only relevant routines in init **TODO:** describe

```
\def\eql@numbering@measure@line@begin{%
  \ifnum\eql@numbering@target@<\z@
    \global\eql@numbering@eqnswinit
  \fi
}
```

TODO: describe

```
\def\eql@numbering@measure@eval{%
  \let\eql@numbering@best@use\eql@false
  \eql@numbering@measure@call
  \ifnum\eql@numbering@target@>\eql@totalrows@
    \eql@numbering@target@\eql@totalrows@
  \fi
  \ifnum\eql@numbering@target@>\z@
    \if@eqnsw\else
      \eql@numbering@target@\z@
    \fi
  \fi
  \ifnum\eql@numbering@target@<\@ne
    \ifdefined\eql@nextlabel
      \eql@warn@label@unused
      \let\eql@nextlabel\@undefined
    \fi
    \ifdefined\eql@nexttag
      \eql@warn@tag@unused
      \let\eql@nexttag\@undefined
    \fi
    \ifdefined\eql@nextlabelname
      \eql@warn@labelname@unused
      \let\eql@nextlabelname\@undefined
    \fi
  \fi
}
```

Multi-Line Print Pass. **TODO:** describe

```
\def\eql@numbering@print@init{%
  \ifnum\eql@numbering@target@<\z@
    \let\eql@numbering@warn\eql@false
    \let\label\eql@label
    \let\tag\eql@tag
    \let\raisetag\eql@raisetag
    \let\numberhere\donumber
    \let\numbernext\nonumber
    \let\eql@nextlabel\eql@blocklabel
    \let\eql@nextlabelname\eql@blocklabelname
    \let\eql@nexttag\eql@blocktag
  \else
    \let\label\eql@label@gobble
    \let\tag\eql@tag@gobble
    \let\raisetag\eql@raisetag@gobble
    \let\numberhere\@empty
    \let\numbernext\@empty
  \fi
}
```

TODO: might select only relevant routines in init **TODO:** describe

```
\def\eql@numbering@print@block@begin{%
  \ifnum\eql@numbering@target@>\z@
    \eql@numbering@makeblockanchor
  \fi
  \ifdefined\eql@numbering@subeq@use
    \eql@numbering@printsubeqlabel
  \fi
}
```

TODO: describe

```
\def\eql@numbering@print@line@begin{%
  \ifnum\eql@numbering@target@<\z@
    \global\eql@numbering@eqnswinit
    \global\eql@raisetag@amount@\z@
  \fi
}
```

TODO: describe

```
\def\eql@numbering@print@line@eval{%
  \ifnum\eql@numbering@target@<\z@\else
    \ifnum\eql@numbering@target@=\eql@row@
      \global\@eqnswtrue
    \else
      \global\@eqnswfalse
    \fi
  \fi
}
```

E.5 Component Display

Showkeys Integration. **TODO:** describe

```
\let\eql@SK@loaded\eql@false
\let\eql@SK@label\@gobble
\let\eql@SK@clearlabel\@empty
\let\eql@SK@setlabel\@gobble
\let\eql@SK@printlabel@right\@empty
\let\eql@SK@printlabel@left\@empty
\let\eql@SK@printlabel@line\@empty
\def\eql@label@clean{\eql@label@org}
\AddToHook{package/showkeys/after}{
  \let\eql@SK@loaded\eql@true
  \def\eql@SK@label#1{\SK@\SK@@label#1}
  \def\eql@SK@clearlabel{\let\eql@SK@lab\relax}
  \eql@SK@clearlabel
  \def\eql@SK@@label#1>#2\SK@{%
    \def\eql@SK@lab{\smash{\SK@labelcolor\showkeyslabelformat{#2}}}%
  }
  \def\eql@SK@setlabel#1{\SK@\eql@SK@@label#1}
  \def\eql@SK@printlabel@right{%
    \ifx\eql@SK@lab\relax\else
      \rlap{\kern\marginparsep\eql@SK@lab}%
      \eql@SK@clearlabel
    \fi
  }
  \def\eql@SK@printlabel@left{%
    \ifx\eql@SK@lab\relax\else
      \llap{\eql@SK@lab\kern\marginparsep}%
      \eql@SK@clearlabel
    \fi
  }
  \def\eql@SK@printlabel@line{%
    \ifx\eql@SK@lab\relax\else
      \dimen@\prevdepth
      \nointerlineskip
      \ifdefined\eql@tagsleft
        \llap{%
          \eql@SK@lab
          \kern\marginparsep
        }%
        \eql@SK@clearlabel
      \else
        \rlap{%
          \dimen@\displaywidth
          \advance\dimen@\marginparsep
          \kern\dimen@
          \eql@SK@lab
        }%
      \fi
      \eql@SK@clearlabel
      \prevdepth\dimen@
    \fi
  }
  \let\eql@label@org\label
  \def\eql@label@clean{\let\SK@\@gobbletwo\eql@label@org}
}
```

Labels.

\eql@compose@label **TODO:** describe

```
\def\eql@compose@label{%
  \eql@SK@clearlabel
  \ifdefined\eql@nextlabel
    \ifnum
      \ifnum\eql@numbering@target@<\z@
        \eql@row@
      \else
        \eql@numbering@target@
      \fi=\eql@row@
      \eql@compose@label@
    \fi
  \fi
}
```

`\eql@compose@label@` **TODO:** describe

```
\def\eql@compose@label@{%
  \eql@SK@setlabel\eql@nextlabel
  \begingroup
    \ifnum\eql@numbering@target@=\eql@row@
      \eql@numbering@setblockanchor
    \fi
    \ifdefined\eql@nextlabelname
      \let\@currentlabelname\eql@nextlabelname
      \let\eql@nextlabelname\@undefined
    \else
      \let\@currentlabelname\eql@labelname@generic
    \fi
    \expandafter\eql@label@clean\expandafter{\eql@nextlabel}%
    \global\let\eql@nextlabel\@undefined
  \endgroup
}
```

TODO: describe

```
\def\eql@numbering@printsubeqlabel{%
  \ifdefined\eql@parentlabel
    \eql@numbering@makeblockanchor
    \eql@SK@setlabel\eql@parentlabel
    \begingroup
      \def\@currentcounter{equation}%
      \eql@numbering@setblockanchor
      \let\@currentlabelname\eql@labelname@generic
      \protected@edef\@currentlabel{\p@equation\theparentequation}%
      \expandafter\eql@label@clean\expandafter{\eql@parentlabel}%
    \endgroup
    \eql@SK@printlabel@line
  \fi
}
```

Hyperref Anchors. **TODO:** describe

```
\let\eql@Hy@anchor\@gobble
\AddToHook{package/hyperref/after}{
  \def\eql@Hy@anchor#1{%
    \Hy@raisedlink{\hyper@anchor{#1}}%
  }%
}
```

TODO: describe

```
\def\eql@numbering@makeblockanchor{%
  \eql@numbering@refstep
  \global\edef\eql@label@currentHref{\eql@numbering@ref}%
  \eql@Hy@anchor\eql@label@currentHref
  \global\edef\eql@label@thepage{\thepage}%
}
\def\eql@numbering@setblockanchor{%
  \let\thepage\eql@label@thepage
  \let\@currentHref\eql@label@currentHref
}
```

TODO: describe

\eql@compose@anchor

```
\def\eql@compose@anchor{%
  \ifdefined\eql@nexttag
    \expandafter\eql@tag@eval\eql@nexttag
    \def\@currentcounter{equation}%
    \let\@currentlabel\eql@tag@label
    \eql@numbering@refstep
    \edef\@currentHref{\eql@numbering@ref}%
    \eql@Hy@anchor\@currentHref
    \global\let\eql@nexttag\@undefined
  \else
    \refstepcounter{equation}%
    \let\eql@tag@tool\eql@tag@form
    \edef\eql@tag@text{\theequation}%
  \fi
}
```

Tag Layout. **TODO:** describe

```
\def\eql@tag@setbox@#1{%
  \def\eql@tag@box##1{#1}%
}
\def\eql@tag@setbox#1{%
  \def\eql@tag@box##1{\hbox{\m@th\normalfont#1}}%
}
```

TODO: describe

```
\def\eql@tag@setform@#1{%
  \def\eql@tag@form##1{#1}%
}
\def\eql@tag@setform#1#2#3{%
  \def\eql@tag@form##1{#1\ignorespaces#2\unskip\@@italiccorr#3}%
}
```

```
\eql@tag@setbox{#1}
\eql@tag@setform({#1})
\def\eql@tag@boxedform#1{\eql@tag@box{\eql@tag@form{#1}}}
```

```
\protected\def\tagform{\eql@tag@form}
\protected\def\tagbox{\eql@tag@box}
\protected\def\tagboxed{\eql@tag@boxedform}
```

\eqref amsmath defines the macro \eqref to refer to equation labels in a proper format. We provide it for completeness:

```
\protected\def\eql@eqref#1{\textup{\eql@tag@boxedform{\ref{#1}}}}
```

\eql@compose@tag **TODO:** describe

```
\def\eql@compose@tag{%
  \eql@tagging@tagbegin
  \eql@tag@box{%
    \eql@tag@tool\eql@tag@text
    \eql@tagging@tagsave
  }%
  \eql@tagging@tagend
}
```

E.6 Tag Composition

TODO: describe

```
\def\eql@compose@measure{%
  \ifdefined\eql@nexttag
    \expandafter\eql@tag@eval\eql@nexttag
    \eql@tag@box{\eql@tag@tool\eql@tag@text}%
  \else
    \stepcounter{equation}%
    \eql@tag@box{\eql@tag@form\theequation}%
  \fi
  \ifnum\eql@numbering@target@<\z@
    \global\let\eql@nextlabel\@undefined
    \global\let\eql@nexttag\@undefined
  \fi
}
```

TODO: describe

```
\def\eql@compose@print{%
  \eql@compose@anchor
  \eql@compose@label
  \ifdefined\eql@tagsleft
    \eql@SK@printlabel@left
    \eql@compose@tag
  \else
    \eql@compose@tag
    \eql@SK@printlabel@right
  \fi
}
```

TODO: describe

TODO: one might still compare width to zero and pretend the tag is absent??

```
\def\eql@tagbox@make#1{%
  \setbox\eql@tagbox@\hbox{\eql@strut@tag\@lign#1}%
  \eql@tagwidth@\wd\eql@tagbox@
  \ifdim\eql@tagwidth@<\eql@tagwidthmin@
    \eql@tagwidth@\eql@tagwidthmin@
  \fi
  \advance\eql@tagwidth@\eql@tagsepmin@
}
```

TODO: describe

```
\def\eql@tagbox@print@tagsright{%
  \kern-\wd\eql@tagbox@
  \box\eql@tagbox@
}
```

TODO: describe

```
\def\eql@tagbox@print@tagsleft{%
  \wd\eql@tagbox@\z@
  \box\eql@tagbox@
}
```

TODO: describe

```
\def\eql@tagbox@print@tagsright@raise{%
```

```
  \ifnum\eql@row@=\eql@totalrows@
    \global\advance\eql@raisetag@firstlast@\tw@
  \fi
  \llap{\vtop{%
    \vskip-\eql@raisetag@amount@
    \normalbaselines
    \setbox\@ne\hbox{}%
    \dp\@ne\eql@line@depth@
    \box\@ne
    \box\eql@tagbox@
  }}%
}
```

TODO: describe

```
\def\eql@tagbox@print@tagsleft@raise{%
  \ifnum\eql@row@=\@ne
    \global\advance\eql@raisetag@firstlast@\@ne
  \fi
  \rlap{\vbox{%
    \normalbaselines
    \box\eql@tagbox@
    \vbox to\eql@line@height@{}%
    \vskip\eql@raisetag@amount@
  }}%
}
```

ql@tagbox@print@cell

```
\def\eql@tagbox@print@cell{%
  \advance\eql@tagwidth@-\eql@tagfuzz@
  \ifdefined\eql@tagsleft
    \eql@display@firstavail@set\z@
    \ifdim\eql@tagwidth@>\eql@line@avail@
      \eql@tagbox@print@tagsleft@raise
    \else
      \eql@tagbox@print@tagsleft
    \fi
    \kern\displaywidth
  \else
    \kern\displaywidth
    \advance\eql@tagwidth@\eql@line@width@
    \ifdim\eql@tagwidth@>\displaywidth
      \eql@tagbox@print@tagsright@raise
    \else
      \eql@tagbox@print@tagsright
    \fi
  \fi
}
```

F Subequation Numbering

We replicate the amsmath functionality to number a block of equations with a common number and a sub-numbering.

F.1 Definitions

entequation (*counter*) We define a counter to store the main equation number while in subequation mode. It makes sense to share this definition with amsmath as parentequation, and we need to undefine it when amsmath is loaded at a later stage:

```
\eql@amsmath@undefine\c@parentequation
\eql@amsmath@undefine\theparentequation
\ifdefined\c@parentequation\else
\newcounter{parentequation}
\fi
```

ubequations@template We store a template which will installed as \theequation in subequations mode: **TODO:** need to protect something?!

```
\def\eql@subequations@template{\theparentequation\alph{equation}}
```

@subequations@active A boolean register which tells whether subequations are in use and thus must not be invoked again:

```
\let\eql@subequations@active\eql@false
```

ql@subequations@init Low-level initialise the subequations mode. Store the equation counter in \eql@subequations@restorecounter for the case that no equation numbers will be used. Step the equation counter, copy it to parentequation and initialise \theparentequation (and its hyperref counterpart) with the expanded current value of \theequation; fill with tag data instead if a tag has been specified. Reset the equation counter and use the template for \thequation:

```
\def\eql@subequations@init{%
  \edef\eql@subequations@restorecounter{%
    \global\c@equation\the\c@equation\relax}%
  \ifdefined\eql@blocktag
    \expandafter\eql@tag@eval\eql@blocktag
    \eql@numbering@refstep
    \protected@edef\theHparentequation{\eql@numbering@ref}%
    \protected@edef\theparentequation{\eql@tag@text}%
  \else
    \advance\c@equation\@ne
    \protected@edef\theparentequation{\theequation}%
    \ifdefined\theHequation
      \protected@edef\theHparentequation{\theHequation}%
    \fi
  \fi
  \global\c@parentequation\c@equation
  \global\c@equation\z@
  \let\theequation\eql@subequations@template
  \def\theHequation{\theHparentequation.\arabic{equation}}%
}
```

l@subequations@close Low-level close the subequations mode. If no number has been used, return to the original equation counter, otherwise use the value stored in parentequation. Note that we cannot use \setcounter here because the calc version would involve actions which are not allowed after \halign within a display equation:

```
\def\eql@subequations@close{%
  \ifnum\c@equation=\z@
    \eql@subequations@restorecounter
  \else
    \global\c@equation\c@parentequation
  \fi
}
```

F.2 Environment

l@subequations@start Start the subequations environment with optional parameters in `#1`. Enter subequations mode and set an anchor for subsequent `\label`'s. Manually print the showkeys tag: **TODO:** join with other similar anchor routines `\eql@numbering@printsubeqlabel`

```
\def\eql@subequations@start{%
  \let\eql@blocklabel\@undefined
  \let\eql@blocklabelname\@undefined
  \let\eql@blocktag\@undefined
  \eql@nextopt@process{subequations}%
  \eql@subequations@init
  \eql@numbering@refstep
  \edef\eql@subequations@currentHref{\eql@numbering@ref}%
  \eql@Hy@anchor\eql@subequations@currentHref
  \edef\eql@subequations@thepage{\thepage}%
  \def\@currentcounter{equation}%
  \let\@currentHref\eql@subequations@currentHref
  \protected@edef\@currentlabel{\p@equation\theparentequation}%
  \ifdefined\eql@blocklabelname
    \let\@currentlabelname\eql@blocklabelname
  \else
    \let\@currentlabelname\eql@labelname@generic
  \fi
  \let\eql@subequations@active\eql@true
  \ifdefined\eql@blocklabel
    \eql@SK@label\eql@blocklabel
  \fi
  \ignorespaces
}
```

eql@subequations@end End the subequations environment. Issue the label if one has been specified and an equation number has actually been used. Then close subequations mode:

```
\def\eql@subequations@end{%
  \ifnum\c@equation>\z@
    \ifdefined\eql@blocklabel
      \begingroup
        \def\@currentcounter{equation}%
        \let\thepage\eql@subequations@thepage
        \let\@currentHref\eql@subequations@currentHref
% \TODO how about tag* ?! also within equations!
        \protected@edef\@currentlabel{\p@equation\theparentequation}%
        \ifdefined\eql@blocklabelname
          \let\@currentlabelname\eql@blocklabelname
        \else
          \let\@currentlabelname\eql@labelname@generic
        \fi
        \expandafter\eql@label@clean\expandafter{\eql@blocklabel}%
      \endgroup
    \fi
  \fi
  \eql@subequations@close
  \ignorespacesafterend
}
```

`subequations` (*env.*) The subequations environment tests for optional parameters and passes on to the start and end routines:

```
\newenvironment{eql@subequations}{%
⟨dev⟩\eql@dev@enterenv
  \eql@subequations@testall\eql@subequations@start%
}{%
  \eql@subequations@end
⟨dev⟩\eql@dev@leaveenv
}
```

TODO: describe

```
\def\eql@subequations@testall{\eql@subequations@testopt}
\def\eql@subequations@testopt#1{%
  \eql@ifnextchar@tight[%]
    {\eql@subequations@addopt{\eql@subequations@testat{#1}}}%
    {\eql@subequations@testat{#1}}}
\def\eql@subequations@addopt#1[#2]{\eqnaddopt{#2}#1}
\def\eql@subequations@testat#1{%
  \eql@ifat@tight%
    {\eql@subequations@addlabel{#1}}%
    {#1}}
\def\eql@subequations@addlabel#1#2{\eqnaddopt{label={#2}}#1}
```

F.3 Subequation Scheme

TODO: describe

```
\def\eql@numbering@subeq@init{%
  \let\eql@save@theequation\theequation
  \let\eql@save@theHequation\theHequation
  \eql@subequations@init
  \let\eql@parentlabel\eql@blocklabel
  \let\eql@parentlabelname\eql@blocklabelname
  \let\eql@parenttag\eql@blocktag
  \let\eql@blocklabel\@undefined
  \let\eql@blocklabelname\@undefined
  \let\eql@blocktag\@undefined
}
```

TODO: describe

```
\def\eql@numbering@subeq@test{%
  \ifnum\c@equation<\tw@
    \let\eql@numbering@subeq@use\@ne
  \fi
}
```

TODO: describe

```
\def\eql@numbering@subeq@revert{%
  \let\eql@blocklabel\eql@parentlabel
  \let\eql@blocklabelname\eql@parentlabelname
  \let\eql@blocktag\eql@parenttag
  \let\eql@numbering@subeq@use\eql@false
  \let\theequation\eql@save@theequation
  \let\theHequation\eql@save@theHequation
  \eql@subequations@restorecounter
}
```

TODO: describe

```
% \TODO note must not use setcounter here (when calc is loaded)
\def\eql@numbering@subeq@close{%
  \eql@subequations@close
}
```

G Display Equations Support

TODO: describe

G.1 Display Breaks

TODO: describe

erdisplaylinepenalty

```
\interdisplaylinepenalty\@M
```

\eql@getdsp@pen **TODO:** isn't this the opposite order than `\@getpen`?!

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

TODO: allow a displaybreak before equations

```
\protected\def\eql@displaybreak@default{%
  \eql@warning{Invalid use of \string\displaybreak}{}%
  \eql@teststaroropt@loose\@gobble\eql@gobbleopt{}}
\eql@amsmath@after{\let\eql@displaybreak@default\displaybreak}
\eql@amsmath@let\displaybreak\eql@displaybreak@default

\newcount\eql@displaybreak@pen@
\newcount\eql@displaybreak@prepen@
\newcount\eql@displaybreak@postpen@
```

TODO: describe

```
\protected\def\eql@displaybreak{%
  \eql@ampprotecttwo\eql@teststaroropt@tight
    \eql@displaybreak@star\eql@displaybreak@opt{4}%
}

\def\eql@displaybreak@star#1{%
  \global\eql@displaybreak@pen@\numexpr#1\relax
}

\def\eql@displaybreak@opt[#1]{%
  \ifnum#1<\z@
    \global\eql@displaybreak@pen@\@MM
  \else
    \global\eql@displaybreak@pen@-\@getpen{#1}\relax
  \fi
}
```

TODO: describe

```
\def\eql@displaybreak@pre#1{%
  \ifnum#1<\z@
    \eql@displaybreak@prepen@\@MM
  \else
    \eql@displaybreak@prepen@-\@getpen{#1}\relax
  \fi
}
```

TODO: describe

```
\def\eql@displaybreak@post#1{%
  \ifnum#1<\z@
    \eql@displaybreak@postpen@\@MM
  \else
    \eql@displaybreak@postpen@-\@getpen{#1}\relax
  \fi
}
```

TODO: describe

```
\def\eql@displaybreak@inter#1{%
  \ifnum#1<\z@
    \interdisplaylinepenalty\@M
  \else
    \interdisplaylinepenalty-\eql@getdsp@pen{#1}\relax
  \fi
}
```

G.2 Vertical Spacing Support

TODO: describe

`\eql@strut` `\eql@strutbox@`

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

```
\newbox\eql@strutbox@
\def\eql@strut@depth{.3}
\def\eql@strut{\copy\eql@strutbox@}
\let\eql@strut@cell\eql@strut
\let\eql@strut@tag\eql@strut
\def\eql@strut@make{%
  \setbox\eql@strutbox@\hbox{%
    \@tempdimb\dimexpr
        \eql@strut@depth\normalbaselineskip+.5\normallineskiplimit\relax
    \@tempdima\dimexpr
        \normalbaselineskip-\normallineskiplimit-\@tempdimb\relax
    \vrule\@height\@tempdima\@depth\@tempdimb\@width\z@
  }
}
\AtBeginDocument{\eql@strut@make}
```

TODO: describe

```
\def\eql@spread@set{%
  \eql@spread@\dimexpr\glueexpr\eql@spread@val\relax
```

```
      +\normalbaselineskip-\baselineskip\relax
  \ifdim\eql@spread@>\z@
    \openup\eql@spread@
    \ifdefined\spread@equation
      \let\spread@equation\@empty
    \fi
  \fi
}
```

G.3 Entry and Exit

TODO: describe

eql@vspaceskip@ (*skip*)
eql@abovespace@ (*skip*)
eql@belowspace@ (*skip*)

TODO: add a proper star variant?!

```
\newskip\eql@vspaceskip@
\newskip\eql@abovespace@
\newskip\eql@belowspace@
\let\eql@vspace@org\vspace
\def\eql@vspace{\eql@ifstar@loose\eql@vspace@\eql@vspace@}
\def\eql@vspace@#1{%
  \global\advance\eql@vspaceskip@\glueexpr#1\relax}
```

```
\def\eql@display@enter{%
  \ifvmode
    \eql@prevdepth@\prevdepth
    \nointerlineskip
    \noindent
  \else
    \eql@prevdepth@\maxdimen
  \fi
}
```

\eql@display@adjust

```
\def\eql@display@adjust{%
  \ifdefined\eql@display@linewidth
    \displaywidth\glueexpr\eql@display@linewidth\relax
    \advance\displaywidth-\displayindent
  \fi
  \ifdefined\eql@display@marginleft
    \advance\displaywidth\displayindent
    \displayindent\glueexpr\eql@display@marginleft\relax
    \advance\displaywidth-\displayindent
  \fi
  \ifdefined\eql@display@marginright
    \advance\displaywidth-\glueexpr\eql@display@marginright\relax
  \fi
  \ifdim\displaywidth<\z@
    \displaywidth\z@
  \fi
}
```

\eql@display@init

```
\def\eql@display@init{%
  \let\displaybreak\eql@displaybreak
  \let\eql@vspace@org\vspace
  \let\vspace\eql@vspace
  \eql@spread@set
  \eql@strut@make
}
```

\eql@display@print

```
\def\eql@display@print{%
  \eql@display@firstavail@\z@
  \eql@raisetag@firstlast@\z@
  \eql@displaybreak@pen@\@MM
  \eql@vspaceskip@\z@skip
}
```

@display@halign@init

TODO: describe

```
\def\eql@display@halign@init#1{%
  \eql@row@\z@
  \eql@prevgraf@\prevgraf
  \everycr{\noalign{%
    \global\advance\eql@row@\@ne
    \prevgraf\numexpr\prevgraf+\@ne\relax
    #1%
  }}%
}
```

TODO: how about penalty here? not for entry into display

```
\def\eql@display@halign@start{%
  \prevgraf\numexpr\eql@prevgraf@+\@ne\relax
  \ifdim\eql@prevdepth@=\maxdimen\else
    \prevdepth\eql@prevdepth@
  \fi
  \ifdim\prevdepth=-\@m\p@\else
    \ifdefined\eql@display@height
      \skip@\baselineskip
      \advance\skip@-\glueexpr\eql@display@height\relax
      \advance\skip@-\prevdepth\relax
      \ifdim\skip@<\lineskiplimit
        \skip@\lineskip
      \fi
      \advance\skip@-\eql@spread@\relax
      \vskip\skip@
      \nointerlineskip
    \else
      \vskip-\eql@spread@\relax
    \fi
  \fi
}
```

TODO: describe

```
\def\eql@display@vspace{%
  \advance\abovedisplayskip\eql@abovespace@
  \advance\belowdisplayskip\eql@belowspace@
  \advance\belowdisplayskip\eql@vspaceskip@
}
```

TODO: describe

```
\def\eql@display@vspace@native{%
```

```
  \advance\abovedisplayskip\eql@abovespace@
  \advance\belowdisplayskip\eql@belowspace@
  \advance\belowdisplayskip\eql@vspaceskip@
  \advance\abovedisplayshortskip\eql@abovespace@
  \advance\belowdisplayshortskip\eql@belowspace@
  \advance\belowdisplayshortskip\eql@vspaceskip@
}
```

TODO: describe

```
\def\eql@display@penalty{%
  \ifnum\eql@displaybreak@postpen@=\@MM\else
    \postdisplaypenalty\eql@displaybreak@postpen@
  \fi
  \ifnum\eql@displaybreak@pen@=\@MM\else
    \postdisplaypenalty\eql@displaybreak@pen@
  \fi
  \ifnum\eql@displaybreak@prepen@=\@MM\else
    \predisplaypenalty\eql@displaybreak@prepen@
  \fi
}
```

TODO: describe

```
\def\eql@display@halign@end{%
  \global\eql@prevgraf@\numexpr\prevgraf+\@ne\relax
  \ifdefined\eql@display@depth
    \prevdepth\glueexpr\eql@display@depth\relax
  \fi
}
```

`\eql@display@close` **TODO:** there seems to be an offset of 1em in predisplaysize towards actual content, nice. **TODO:** must not use setlength or setcounter when calc is loaded **TODO:** do we actually need penalty adjustments in case of paragraphs above or below?

```
\def\eql@display@close{%
% \TODO temporary fix for development stages
  \ifdefined\eql@tagging@on
    \ifdefined\dollardollar@begin\else
      \belowdisplayskip-\belowdisplayskip
      \belowdisplayshortskip-\belowdisplayshortskip
    \fi
  \fi
  \ifdim\eql@display@firstavail@<\z@
    \eql@display@firstavail@\z@
  \fi
  \eql@skip@mode@leave@\z@
  \ifdim\eql@prevdepth@=\maxdimen
    \ifdim\predisplaysize=-\maxdimen
      \eql@skip@mode@above@\eql@skip@mode@cont@above\relax
      \eql@skip@mode@below@\eql@skip@mode@cont@below\relax
    \else
      \eql@skip@mode@above@\z@
      \eql@skip@mode@below@\z@
      \advance\eql@display@firstavail@\displayindent
      \ifdim\eql@display@firstavail@>\predisplaysize
        \ifcase\eql@skip@mode@short\relax
        \or
          \eql@skip@mode@above@\@ne
        \or
          \eql@skip@mode@above@\@ne
          \ifnum\eql@totalrows@=\@ne
            \eql@skip@mode@below@\@ne
          \fi
        \or
          \eql@skip@mode@above@\@ne
          \eql@skip@mode@below@\@ne
        \fi
      \fi
    \fi
  \else
    \ifdim\eql@prevdepth@=-\@m\p@
      \eql@skip@mode@above@\eql@skip@mode@top@above\relax
      \eql@skip@mode@below@\eql@skip@mode@top@below\relax
    \else
      \eql@skip@mode@above@\eql@skip@mode@par@above\relax
      \eql@skip@mode@below@\eql@skip@mode@par@below\relax
    \fi
  \fi
  \ifcase\eql@skip@mode@above@
  \or\or\or
    \predisplaypenalty\z@
  \or
    \predisplaypenalty\z@
  \fi
  \ifcase\eql@skip@mode@below@
  \or\or\or
    \eql@skip@mode@leave@\@ne
  \or
    \eql@skip@mode@leave@\tw@
  \fi
  \ifdefined\eql@skip@force@above
    \eql@skip@mode@above@\eql@skip@force@above\relax
  \fi
  \ifdefined\eql@skip@force@below
    \eql@skip@mode@below@\eql@skip@force@below\relax
  \fi
  \ifdefined\eql@skip@force@leave
    \eql@skip@mode@leave@\eql@skip@force@leave\relax
  \fi
  \ifnum\eql@skip@mode@leave@>\z@
    \postdisplaypenalty\z@
  \fi
  \ifodd\eql@raisetag@firstlast@
    \ifcase\eql@skip@mode@above@
      \abovedisplayskip\glueexpr\eql@skip@tag@above\relax
    \or
      \abovedisplayskip\glueexpr\eql@skip@tag@above\relax
    \or
      \abovedisplayskip\glueexpr\eql@skip@tag@above\relax
    \or
      \abovedisplayskip\glueexpr\eql@skip@partag@above\relax
    \or
      \abovedisplayskip\glueexpr\eql@skip@partag@above\relax
    \or
      \abovedisplayskip\z@skip
    \or
      \abovedisplayskip\glueexpr\eql@skip@medtag@above\relax
    \or
      \abovedisplayskip\glueexpr\eql@skip@custom@above\relax
    \fi
  \else
    \ifcase\eql@skip@mode@above@
      \abovedisplayskip\glueexpr\eql@skip@long@above\relax
    \or
      \abovedisplayskip\glueexpr\eql@skip@short@above\relax
    \or
      \abovedisplayskip\glueexpr\eql@skip@cont@above\relax
    \or
      \abovedisplayskip\glueexpr\eql@skip@par@above\relax
    \or
      \abovedisplayskip\glueexpr\eql@skip@top@above\relax
    \or
      \abovedisplayskip\z@skip
    \or
      \abovedisplayskip\glueexpr\eql@skip@med@above\relax
    \or
      \abovedisplayskip\glueexpr\eql@skip@custom@above\relax
    \fi
  \fi
  \ifnum\eql@raisetag@firstlast@>\@ne
    \ifcase\eql@skip@mode@below@
      \belowdisplayskip\glueexpr\eql@skip@tag@below\relax
    \or
      \belowdisplayskip\glueexpr\eql@skip@tag@below\relax
    \or
      \belowdisplayskip\glueexpr\eql@skip@tag@below\relax
    \or
      \belowdisplayskip\glueexpr\eql@skip@partag@below\relax
    \or
      \belowdisplayskip\glueexpr\eql@skip@partag@below\relax
    \or
      \belowdisplayskip\z@skip
    \or
      \belowdisplayskip\glueexpr\eql@skip@medtag@below\relax
    \or
      \belowdisplayskip\glueexpr\eql@skip@custom@below\relax
    \fi
  \else
    \ifcase\eql@skip@mode@below@
      \belowdisplayskip\glueexpr\eql@skip@long@below\relax
    \or
      \belowdisplayskip\glueexpr\eql@skip@short@below\relax
    \or
      \belowdisplayskip\glueexpr\eql@skip@cont@below\relax
    \or
      \belowdisplayskip\glueexpr\eql@skip@par@below\relax
    \or
      \belowdisplayskip\glueexpr\eql@skip@top@below\relax
    \or
      \belowdisplayskip\z@skip
    \or
      \belowdisplayskip\glueexpr\eql@skip@med@below\relax
    \or
      \belowdisplayskip\glueexpr\eql@skip@custom@below\relax
    \fi
  \fi
  \global\eql@skip@mode@leave@\eql@skip@mode@leave@
  \eql@display@penalty
  \eql@display@vspace
% \TODO temporary fix for development stages
  \ifdefined\eql@tagging@on
    \ifdefined\dollardollar@begin\else
      \belowdisplayskip-\belowdisplayskip
    \fi
  \fi
}
```

TODO: describe

```
\def\eql@display@leave{%
  \prevgraf\eql@prevgraf@
  \ifcase\eql@skip@mode@leave@
  \or
    \endgraf
  \or
    \endgraf
    \prevdepth-\@m\p@
  \fi
}
```

TODO: describe

```
\def\eql@display@nest{%
  \let\label\eql@label@org
  \let\tag\eql@tag@default
  \let\raisetag\eql@raisetag@default
  \let\displaybreak\eql@displaybreak@default
  \let\intertext\eql@intertext@default
  \let\vspace\eql@vspace@org
}
```

TODO: describe

```
\expandafter\def\expandafter\@arrayparboxrestore\expandafter{%
  \@arrayparboxrestore
  \eql@display@nest
  \ifdefined\eql@ampproof@active
    \eql@amprevert
  \fi
  \@displayfalse
}
```

G.4 Stack

TODO: describe

```
\def\eql@stack@enable{%
  \let\eql@stack@save@single\eql@stack@save@single@
  \let\eql@stack@save@multi\eql@stack@save@multi@
  \let\eql@stack@save@boxed\eql@stack@save@boxed@
}
```

TODO: describe

```
\let\eql@stack@save@single\eql@stack@enable
```

```
\let\eql@stack@save@multi\eql@stack@enable
\let\eql@stack@save@boxed\eql@stack@enable
\let\eql@stack@restore\@empty
```

TODO: describe

```
\def\eql@stack@save@reg#1{\global#1\the#1\relax}
\def\eql@stack@save@let#1#2{\global\let\noexpand#2\noexpand#1}
```

TODO: describe

```
\def\eql@stack@save@single@{%
  \let\eql@stack@nextlabel\eql@nextlabel
  \let\eql@stack@nextlabelname\eql@nextlabelname
  \let\eql@stack@nexttag\eql@nexttag
  \edef\eql@stack@restore{%
    \global\if@eqnsw\noexpand\@eqnswtrue\else\noexpand\@eqnswfalse\fi
    \eql@stack@save@let\eql@stack@nextlabel\eql@nextlabel
    \eql@stack@save@let\eql@stack@nextlabelname\eql@nextlabelname
    \eql@stack@save@let\eql@stack@nexttag\eql@nexttag
    \eql@stack@save@let\eql@stack@frame@cmd\eql@frame@cmd
    \eql@stack@save@reg\eql@displaybreak@pen@
    \eql@stack@save@reg\eql@vspaceskip@
    \eql@stack@save@reg\eql@shape@pos@
    \eql@stack@save@reg\eql@shape@amount@
    \eql@stack@save@reg\eql@display@firstavail@
    \eql@stack@save@reg\eql@raisetag@amount@
    \eql@stack@save@reg\eql@raisetag@firstlast@
    \eql@stack@save@reg\eql@row@
  }%
}
```

TODO: describe

```
\def\eql@stack@save@multi@{%
  \let\eql@stack@nextlabel\eql@nextlabel
  \let\eql@stack@nextlabelname\eql@nextlabelname
  \let\eql@stack@nexttag\eql@nexttag
  \let\eql@stack@widthdata@tab\eql@widthdata@tab
  \let\eql@stack@label@thepage\eql@label@thepage
  \let\eql@stack@label@currentHref\eql@label@currentHref
  \edef\eql@stack@restore{%
    \global\if@eqnsw\noexpand\@eqnswtrue\else\noexpand\@eqnswfalse\fi
    \eql@stack@save@let\eql@stack@nextlabel\eql@nextlabel
    \eql@stack@save@let\eql@stack@nextlabelname\eql@nextlabelname
    \eql@stack@save@let\eql@stack@nexttag\eql@nexttag
    \eql@stack@save@let\eql@stack@widthdata@tab\eql@widthdata@tab
    \eql@stack@save@let\eql@stack@label@thepage\eql@label@thepage
    \eql@stack@save@let\eql@stack@label@currentHref\eql@label@currentHref
    \eql@stack@save@let\eql@stack@frame@cmd\eql@frame@cmd
    \eql@stack@save@reg\eql@displaybreak@pen@
    \eql@stack@save@reg\eql@vspaceskip@
    \eql@stack@save@reg\eql@shape@pos@
    \eql@stack@save@reg\eql@shape@amount@
    \eql@stack@save@reg\eql@display@firstavail@
    \eql@stack@save@reg\eql@raisetag@amount@
    \eql@stack@save@reg\eql@raisetag@firstlast@
    \eql@stack@save@reg\eql@column@
    \eql@stack@save@reg\eql@totalcolumns@
    \eql@stack@save@reg\eql@line@avail@
    \eql@stack@save@reg\eql@line@pos@
    \eql@stack@save@reg\eql@line@width@
    \eql@stack@save@reg\eql@line@depth@
    \eql@stack@save@reg\eql@line@height@
    \eql@stack@save@reg\eql@tagwidth@max@
    \eql@stack@save@reg\eql@numbering@target@
    \eql@stack@save@reg\eql@row@
    \eql@stack@save@reg\eql@tagrows@
  }%
}
\def\eql@stack@save@boxed@{%
  \edef\eql@stack@restore{%
    \eql@stack@save@reg\eql@row@
    \eql@stack@save@reg\eql@totalrows@
    \eql@stack@save@reg\eql@shape@pos@
    \eql@stack@save@reg\eql@shape@amount@
    \eql@stack@save@let\eql@stack@frame@cmd\eql@frame@cmd
  }%
}
```

H Multi-Line Support

TODO: describe

H.1 Measure Support

TODO: describe

```
\def\eql@measure@init#1#2{%
  \eql@widthdata@reset
  \eql@numbering@measure@init
  \eql@row@\z@
  \tabskip\z@skip
  \everycr{\noalign{%
    \global\advance\eql@row@\@ne
    #1%
  }}%
  \eql@measure@savestate
  \measuring@true
  \eql@display@halign@letcr{#2}%
}
```

```
\def\eql@measure@close{%
  \advance\eql@row@-\tw@
  \eql@totalrows@\eql@row@
  \eql@numbering@measure@eval
  \ifnum\eql@numbering@target@>\z@
    \eql@tagbox@make\eql@compose@measure
    \eql@widthdata@savetagsingle
  \fi
  \eql@measure@restorestate
}
```

measure@restorestate
ql@measure@savestate

```
\let\eql@measure@restorestate\@empty
\def\eql@measure@savestate{%
```

```
  \begingroup
    \def\@elt##1{%
      \global\csname c@##1\endcsname\the\csname c@##1\endcsname}%
    \global\edef\@gtempa{%
      \cl@@ckpt
      \let\noexpand\eql@measure@restorestate\noexpand\@empty
      \ifmeasuring@\noexpand\measuring@true\else\noexpand\measuring@false\fi
    }%
  \endgroup
  \let\eql@measure@restorestate\@gtempa
}
```

H.2 Line Breaks

TODO: describe

\eql@display@cr

```
\protected\def\eql@display@cr{%
```

```
  \eql@ampprotecttwo\eql@teststaropt@tight
    {\global\eql@displaybreak@pen@\@M\eql@display@cr@opt}
    \eql@display@cr@opt\z@
}
```

\eql@display@cr@opt

```
\def\eql@display@cr@opt[#1]{%
  \eql@display@endline
  \cr
```

```
  \noalign{%
    \ifnum\eql@displaybreak@pen@=\@MM
      \penalty\interdisplaylinepenalty
      \global\eql@displaybreak@pen@\@MM
    \else
      \penalty\eql@displaybreak@pen@
    \fi
    \advance\eql@vspaceskip@\glueexpr#1\relax%
    \vskip\eql@vspaceskip@
    \global\eql@vspaceskip@\z@skip
  }%
}
```

display@halign@letcr

```
\def\eql@display@halign@letcr#1{%
  \let\\\eql@display@cr
  \let\eql@display@endline#1%
}
```

H.3 Intertext

TODO: describe

TODO: revert in everymath?

```
\def\eql@intertext@default{\eql@error{Invalid use of \string\intertext}}
\eql@amsmath@let\intertext\eql@intertext@default
```

TODO: why does it fail in measuring? total width?! determine total width otherwise!?

```
\def\eql@intertext@process{%
  \eql@display@endline
  \cr
  \ifmeasuring@
    \expandafter\@gobble
  \else
    \expandafter\eql@intertext@print
  \fi
}
```

TODO: describe **TODO:** prevdepth **TODO:** does this have to be in a vbox? **TODO:** vskip and penalty opposite order **TODO:** can we handle short? certainly needs two passes

```
\def\eql@intertext@print#1{%
  \noalign{%
    \eql@display@halign@end
    \let\eql@skip@force@below\z@
    \let\eql@skip@force@above\z@
    \eql@setkeys{intertext}\eql@intertext@opt
    \openup-\eql@spread@
    \penalty\postdisplaypenalty
    \ifcase\eql@skip@force@below\relax
      \advance\eql@vspaceskip@\glueexpr\eql@skip@long@below\relax
    \or
      \advance\eql@vspaceskip@\glueexpr\eql@skip@short@below\relax
    \or
      \advance\eql@vspaceskip@\glueexpr\eql@skip@cont@below\relax
    \or
      \advance\eql@vspaceskip@\glueexpr\eql@skip@par@below\relax
    \or
      \advance\eql@vspaceskip@\glueexpr\eql@skip@top@below\relax
    \or
      \advance\eql@vspaceskip@\z@skip
    \or
      \advance\eql@vspaceskip@\glueexpr\eql@skip@med@below\relax
    \or
      \advance\eql@vspaceskip@\glueexpr\eql@skip@custom@below\relax
    \fi
    \vskip\eql@vspaceskip@
    \global\eql@vspaceskip@\z@skip
    \vbox{%
      \@parboxrestore
      \ifdim
        \ifdim\@totalleftmargin=\z@\linewidth\else-\maxdimen\fi=\columnwidth
      \else
        \parshape\@ne
        \@totalleftmargin\linewidth
      \fi
      \noindent
      \prevgraf\eql@prevgraf@
      \ignorespaces
      #1%
      \par
      \global\eql@prevgraf@\prevgraf
    }%
    \penalty\predisplaypenalty
    \ifcase\eql@skip@force@above\relax
```

```
      \vskip\glueexpr\eql@skip@long@above\relax
    \or
      \vskip\glueexpr\eql@skip@short@above\relax
    \or
      \vskip\glueexpr\eql@skip@cont@above\relax
    \or
      \vskip\glueexpr\eql@skip@par@above\relax
    \or
      \vskip\glueexpr\eql@skip@top@above\relax
    \or
      \vskip\z@skip
    \or
      \vskip\glueexpr\eql@skip@med@above\relax
    \or
      \vskip\glueexpr\eql@skip@custom@above\relax
    \fi
%     \eql@prevdepth@\maxdimen
    \eql@prevdepth@\z@
    \eql@display@halign@start
  }
}
```

TODO: describe

```
\newenvironment{eql@intertext}{%
  \eql@testopt@tight\eql@intertext@{}%
}{%
  \aftergroup\eql@intertext@after
  \ignorespacesafterend
}
```

TODO: describe

```
\def\eql@intertext@env{intertext}
\def\eql@intertext@[#1]{%
  \global\def\eql@intertext@opt{#1}%
  \ifx\@currenvir\eql@intertext@env
    \expandafter\eql@scan@env\expandafter\eql@intertext@inject
  \else
    \expandafter\eql@intertext@process
  \fi
}
```

TODO: describe

```
\def\eql@intertext@inject{%
  \global\edef\eql@intertext@after{%
    \noexpand\eql@intertext@process{%
      \ifx\eql@scan@body\eql@scan@body@dump
        \eql@scan@body@dump
      \else
        \noexpand\scantokens{\eql@scan@body@dump}%
      \fi
    }%
  }%
}
```

I Column Placement

TODO: describe

I.1 Supporting Definitions

l@shape@pos@ (*dimen*)
hape@amount@ (*dimen*)

The registers `\eql@shape@pos@` and `\eql@shape@amount@` specify the currently selected horizontal alignment (`0` for left, `1` for center, `2` for right) and the indentation amount, respectively:

```
\newcount\eql@shape@pos@
\newdimen\eql@shape@amount@
```

l@marginleft@ (*dimen*)
ginleft@min@ (*dimen*)
marginright@ (*dimen*)
enteroffset@ (*dimen*)

The registers `\eql@marginleft@` and `\eql@marginright@` store the intended left and right margin for the equation lines: **TODO:** update

```
\newdimen\eql@marginleft@
\newdimen\eql@marginright@
\newdimen\eql@marginleft@min@
\newdimen\eql@centeroffset@
```

I.2 Shape Schemes

The horizontal alignment of each line is specified by a shape scheme.

`\eql@shape@tab@...` We select the scheme through a `\csname` selector with the following names:

```
\def\eql@shape@tab@default{default}
\def\eql@shape@tab@left{left}
\def\eql@shape@tab@center{center}
\def\eql@shape@tab@right{right}
\def\eql@shape@tab@first{first}
\def\eql@shape@tab@hanging{hanging}
\def\eql@shape@tab@steps{steps}
```

For convenience, we add further alias names for the schemes:

```
\let\eql@shape@tab@def\eql@shape@tab@default
\let\eql@shape@tab@\eql@shape@tab@default
\let\eql@shape@tab@l\eql@shape@tab@left
\let\eql@shape@tab@c\eql@shape@tab@center
\let\eql@shape@tab@r\eql@shape@tab@right
\let\eql@shape@tab@rc\eql@shape@tab@first
\let\eql@shape@tab@indent\eql@shape@tab@first
\let\eql@shape@tab@hang\eql@shape@tab@hanging
\let\eql@shape@tab@lc\eql@shape@tab@hanging
\let\eql@shape@tab@outdent\eql@shape@tab@hanging
\let\eql@shape@tab@lcr\eql@shape@tab@steps
```

`\eql@shape@mode` The currently selected scheme is stored in `\eql@shape@mode`. It it set to `default`:

```
\let\eql@shape@mode\eql@shape@tab@default
```

`\eql@shape@set` Set the scheme via the translation table:

```
\def\eql@shape@set#1{%
  \ifcsname eql@shape@tab@#1\endcsname
```

```
    \expandafter\let\expandafter\eql@shape@mode
      \csname eql@shape@tab@#1\endcsname
  \else
    \eql@error{shape ‘#1’ unknown: setting to default}%
    \let\eql@shape@mode\eql@shape@tab@default
  \fi
}
```

ape@layoutcenter@...
shape@layoutleft@...

Define the uniform shape schemes `left`, `center`, `right` and `default` for the central and left alignment layout. The scheme functions determine the desired alignment and indentation for the current row:

```
\def\eql@shape@layoutcenter@left{\eql@shape@pos@\z@\eql@shape@amount@\z@}
\def\eql@shape@layoutcenter@center{\eql@shape@pos@\@ne\eql@shape@amount@\z@}
\def\eql@shape@layoutcenter@right{\eql@shape@pos@\tw@\eql@shape@amount@\z@}
\let\eql@shape@layoutcenter@default\eql@shape@layoutcenter@center
\def\eql@shape@layoutleft@left{\eql@shape@pos@\z@\eql@shape@amount@\z@}
\def\eql@shape@layoutleft@center{\eql@shape@pos@\@ne\eql@shape@amount@\z@}
\def\eql@shape@layoutleft@right{\eql@shape@pos@\tw@\eql@shape@amount@\z@}
\let\eql@shape@layoutleft@default\eql@shape@layoutleft@left
```

The `first` scheme implements left alignment with indentation for the first line (unless there is only one line):

```
\def\eql@shape@layoutcenter@first{%
  \eql@shape@pos@\z@
  \eql@shape@amount@\z@
  \ifnum\eql@totalrows@>\@ne
    \ifnum\eql@row@=\@ne
      \eql@shape@amount@\eql@indent@
    \fi
  \fi
}
\def\eql@shape@layoutleft@first{%
  \eql@shape@pos@\z@
  \eql@shape@amount@\z@
  \ifnum\eql@totalrows@>\@ne
    \ifnum\eql@row@=\@ne
      \eql@shape@amount@\eql@indent@
    \fi
  \fi
}
```

The `hanging` scheme implements left alignment with hanging indentation for the first line (unless there is only one line). In centeral alignment layout all but the first line are indented while in left aligned layout the first line has negative indentation:

```
\def\eql@shape@layoutcenter@hanging{%
  \eql@shape@pos@\z@
  \eql@shape@amount@\eql@indent@
  \ifnum\eql@totalrows@>\@ne
    \ifnum\eql@row@=\@ne
      \eql@shape@amount@\z@
    \fi
  \fi
}
\def\eql@shape@layoutleft@hanging{%
  \eql@shape@pos@\z@
  \eql@shape@amount@\z@
  \ifnum\eql@totalrows@>\@ne
```

```
    \ifnum\eql@row@=\@ne
      \eql@shape@amount@-\eql@indent@
    \fi
  \fi
}
```

The `steps` scheme implements singles out the first and last lines which are shifted left and right, respectively. In centeral alignment layout the shift operates on the alignment whereas in left alignment layout the shift uses indentation:

```
\def\eql@shape@layoutcenter@steps{%
  \eql@shape@amount@\z@
  \eql@shape@pos@\@ne
  \ifnum\eql@totalrows@>\@ne
    \ifnum\eql@row@=\@ne
      \eql@shape@pos@\z@
    \fi
    \ifnum\eql@row@=\eql@totalrows@
      \eql@shape@pos@\tw@
    \fi
  \fi
}
\def\eql@shape@layoutleft@steps{%
  \eql@shape@pos@\z@
  \eql@shape@amount@\z@
  \ifnum\eql@totalrows@>\@ne
    \ifnum\eql@row@=\@ne
      \eql@shape@amount@-\eql@indent@
    \fi
    \ifnum\eql@row@=\eql@totalrows@
      \eql@shape@amount@\eql@indent@
    \fi
  \fi
}
```

`\eql@shape@select`
`\eql@shape@eval`

Select the shape selector function for the currrent scheme `@\eql@shape@mode` and layout and store it in `\eql@shape@eval`:

```
\let\eql@shape@eval\@undefined
\def\eql@shape@select{%
  \expandafter\let\expandafter\eql@shape@eval
    \csname eql@shape%
    @\ifdefined\eql@layoutleft layoutleft\else layoutcenter\fi
    @\eql@shape@mode\endcsname
}
```

`\eql@shape@alignleft`
`eql@shape@alignright`
`ql@shape@aligncenter`

Adjust the alignment of the current equation line. The optional argument specifies the amount of indentation:

```
\protected\def\eql@shape@alignleft{%
  \global\eql@shape@pos@\z@
  \eql@ampprotect\eql@shape@align@testpar\eql@shape@alignamount@opt}
\protected\def\eql@shape@aligncenter{%
  \global\eql@shape@pos@\@ne
  \eql@ampprotect\eql@shape@align@testpar\eql@shape@alignamount@opt}
\protected\def\eql@shape@alignright{%
  \global\eql@shape@pos@\tw@
  \eql@ampprotect\eql@shape@align@testpar\eql@shape@alignamount@opt}
\def\eql@shape@align@testpar#1{%
```

```
  \eql@ifstar@tight{#1[\eql@indent@]}%
  {\eql@ifnextgobble@tight{!}{#1[-\eql@indent@]}%
  {\eql@testopt@tight{#1}\z@}}}
\def\eql@shape@alignamount@opt[#1]{%
  \global\eql@shape@amount@\glueexpr#1\relax}
```

`ql@shape@alignamount` **TODO:** describe

```
\protected\def\eql@shape@alignamount{%
  \eql@ampprotecttwo\eql@ifstar@tight
    \eql@shape@alignamount@set\eql@shape@alignamount@add}
\def\eql@shape@alignamount@add#1{%
  \ifnum\eql@shape@pos@=\m@ne
    \global\eql@shape@pos@-\thr@@
    \eql@shape@amount@\z@
  \fi
  \global\advance\eql@shape@amount@\glueexpr#1\relax}
\def\eql@shape@alignamount@set#1{%
  \ifnum\eql@shape@pos@<\z@
    \global\eql@shape@pos@-\tw@
  \fi
  \global\eql@shape@amount@\glueexpr#1\relax}
\def\eql@shape@align@enable{%
  \let\shoveleft\eql@shape@alignleft
  \let\shovecenter\eql@shape@aligncenter
  \let\shoveright\eql@shape@alignright
  \let\shoveby\eql@shape@alignamount
}
```

TODO: describe

```
\protected\def\eql@shape@align@default{%
  \eql@warning{\string\shove... not allowed here}%
  \eql@ampprotect\eql@shape@align@testpar\eql@gobbleopt}
\protected\def\eql@shape@alignamount@default{%
  \eql@warning{\string\shove... not allowed here}%
  \eql@ampprotecttwo\eql@ifstar@tight\@gobble\@gobble}
\def\eql@shape@align@disable{%
  \let\shoveleft\eql@shape@align@default
  \let\shovecenter\eql@shape@align@default
  \let\shoveright\eql@shape@align@default
  \let\shoveby\eql@shape@alignamount@default
}
```

I.3 Width Data

`idth@single@` (*dimen*)

```
\newdimen\eql@tagwidth@single@
```

`\eql@widthdata@tab` **TODO:** new

```
\let\eql@widthdata@tab\@empty
```

`\eql@widthdata@reset`

```
\def\eql@widthdata@reset{%
  \let\eql@widthdata@tab\@empty
  \eql@tagwidth@max@\z@
  \eql@tagrows@\z@
}
```

\eql@widthdata@add

```
\def\eql@widthdata@add#1{%
  \expandafter\global\expandafter\def\expandafter\eql@widthdata@tab
    \expandafter{\eql@widthdata@tab#1}%
}
```

l@widthdata@startrow

```
\def\eql@widthdata@startrow{%
  \expandafter\eql@widthdata@add\expandafter{%
    \expandafter\eql@row@\the\eql@row@\relax}%
}
```

l@widthdata@savecell

```
\def\eql@widthdata@savecell{%
  \edef\eql@tmp{%
    \eql@shape@pos@\the\eql@shape@pos@\relax
    \eql@cellwidth@\the\eql@cellwidth@\relax
    \eql@shape@amount@\the\eql@shape@amount@\relax
    \noexpand\eql@widthdata@cellcall
  }%
  \expandafter\eql@widthdata@add\expandafter{\eql@tmp}%
}
```

ql@widthdata@savesep

```
\def\eql@widthdata@savesep{%
  \eql@widthdata@add\eql@widthdata@sepcall
}
```

ql@widthdata@savetag

```
\def\eql@widthdata@savetag{%
  \expandafter\eql@widthdata@add\expandafter{%
    \expandafter\eql@tagwidth@\the\eql@tagwidth@\relax;}%
  \ifdim\eql@tagwidth@>\eql@tagwidth@max@
    \global\eql@tagwidth@max@\eql@tagwidth@
  \fi
  \ifdim\eql@tagwidth@>\z@
    \global\advance\eql@tagrows@\@ne
  \fi
}
```

thdata@savetagsingle

```
\def\eql@widthdata@savetagsingle{%
  \eql@tagwidth@single@\eql@tagwidth@
  \eql@tagwidth@max@\eql@tagwidth@
  \eql@tagrows@\@ne
}
```

\eql@widthdata@for

```
\def\eql@widthdata@for#1{%
```

```
  \def\eql@widthdata@forcall{#1}%
  \expandafter\eql@widthdata@forstep\eql@widthdata@tab
    \eql@row@0\relax\eql@tagwidth@\z@\relax;%
}
```

ql@widthdata@forstep

```
\def\eql@widthdata@forstep\eql@row@#1\relax#2\eql@tagwidth@#3\relax;{%
  \eql@row@#1\relax
  \ifnum\eql@row@=\z@\else
    \eql@tagwidth@#3\relax
    \def\eql@widthdata@cells{#2}%
    \eql@widthdata@forcall
    \expandafter\eql@widthdata@forstep
  \fi
}
```

\eql@widthdata@get

```
\def\eql@widthdata@get#1{%
  \eql@row@#1\relax
  \expandafter\eql@widthdata@getdef\expandafter{\the\eql@row@}%
  \expandafter\eql@widthdata@getparse\eql@widthdata@tab\@nil%
}
```

eql@widthdata@getdef

```
\def\eql@widthdata@getdef#1{%
  \def\eql@widthdata@getparse
    ##1\eql@row@#1\relax##2\eql@tagwidth@##3\relax;##4\@nil{%
    \eql@tagwidth@##3\relax
    \def\eql@widthdata@cells{##2}%
  }%
}
```

\eql@colwidth@tab

```
\let\eql@colwidth@tab\@empty
```

\eql@colwidth@get

```
\def\eql@colwidth@get#1{%
  \ifcase\expandafter#1\eql@colwidth@tab\else\z@\fi
}
```

\eql@colwidth@save

```
\def\eql@colwidth@save#1{%
  \edef\eql@tmp{\noexpand\or\the#1}%
  \expandafter\expandafter\expandafter\def
    \expandafter\expandafter\expandafter\eql@colwidth@tab
    \expandafter\expandafter\expandafter{%
      \expandafter\eql@tmp\eql@colwidth@tab}%
}
```

\eql@widthdata@calc Compute the space that is available at the beginning and at the end of the row stored in `\eql@widthdata@cells`. The space available at the beginning is returned in `\eql@line@avail@`. and `\eql@line@availsep@` describes the number of unused intercolumn separations. The total used width is returned in `\eql@line@width@` and

`\eql@line@widthsep@` describes the number of used intercolumn separations. The available space at the end of the row is given as the difference to `\eql@totalwidth@`:

```
\def\eql@widthdata@calc{%
  \eql@column@\z@
  \eql@line@pos@\z@
  \eql@line@possep@\z@
  \eql@line@avail@\eql@totalwidth@
  \eql@line@availsep@\eql@columns@inter@
  \eql@line@width@\z@
  \eql@line@widthsep@\z@
  \let\eql@widthdata@cellcall\eql@widthdata@calc@call
  \let\eql@widthdata@sepcall\eql@widthdata@calc@callsep
  \eql@widthdata@cells
}
```

`dthdata@calc@callsep` Callback for each intercolumn space.

```
\def\eql@widthdata@calc@callsep{%
  \advance\eql@line@possep@\@ne
}%
```

`@widthdata@calc@call` Callback for each column. When a non-blank cell is encountered, the available space on the left will be fixed if it is still undetermined, and the total width is updated to the current position: **TODO:** implement an offset for central alignment (global?!)

```
\def\eql@widthdata@calc@call{%
  \advance\eql@column@\@ne
  \ifnum\eql@totalcolumns@=\@ne
    \dimen@\eql@totalwidth@
  \else
    \dimen@\eql@colwidth@get\eql@column@\relax
  \fi
  \ifdim\eql@cellwidth@>\z@
    \ifdim\eql@line@width@=\z@
      \eql@line@avail@\eql@line@pos@
      \eql@line@availsep@\eql@line@possep@
      \ifcase\eql@shape@pos@
      \or
        \advance\eql@line@avail@\dimexpr
            (\dimen@-\eql@cellwidth@+\eql@centeroffset@)/\tw@\relax
      \or
        \advance\eql@line@avail@\dimexpr\dimen@-\eql@cellwidth@\relax
      \fi
      \advance\eql@line@avail@\eql@shape@amount@
    \fi
    \eql@line@width@\eql@line@pos@
    \eql@line@widthsep@\eql@line@possep@
    \ifcase\eql@shape@pos@
      \advance\eql@line@width@\eql@cellwidth@
    \or
      \advance\eql@line@width@\dimexpr
          (\dimen@+\eql@cellwidth@+\eql@centeroffset@)/\tw@\relax
    \or
      \advance\eql@line@width@\dimen@
    \fi
    \advance\eql@line@width@\eql@shape@amount@
  \fi
  \advance\eql@line@pos@\dimen@
}
```

I.4 Best Line Selection

@numbering@best@auto **TODO:** describe

```
\let\eql@numbering@best@auto\eql@false
```

g@best@row@ (*counter*)
g@best@space@ (*dimen*)
ering@best@use (*bool*)

```
\newcount\eql@numbering@best@row@
\newdimen\eql@numbering@best@space@
\let\eql@numbering@best@use\eql@false
```

@numbering@best@find Determine the row with the largest available space on the side of the tags:

```
\def\eql@numbering@best@find{%
  \eql@numbering@best@row@\z@
  \eql@numbering@best@space@\z@
  \eql@widthdata@for{%
    \eql@widthdata@calc
    \ifdefined\eql@tagsleft
      \dimen@\eql@line@avail@
    \else
      \dimen@\dimexpr\eql@totalwidth@-\eql@line@width@\relax
    \fi
    \ifdim\dimen@>\eql@numbering@best@space@
      \eql@numbering@best@row@\eql@row@
      \eql@numbering@best@space@\dimen@
    \fi
  }%
  \ifnum\eql@numbering@best@row@>\z@
    \eql@numbering@target@\eql@numbering@best@row@
  \fi
}
```

@numbering@best@test **TODO:** describe

```
\def\eql@numbering@best@test{%
  \eql@widthdata@get\eql@numbering@target@
  \eql@widthdata@calc
  \ifdefined\eql@tagsleft
    \dimen@\dimexpr\eql@line@avail@
        +\eql@marginleft@+\eql@line@availsep@\eql@colsep@\relax
  \else
    \dimen@\dimexpr\displaywidth-\eql@line@width@
        -\eql@marginleft@-\eql@line@widthsep@\eql@colsep@\relax
  \fi
  \ifdim\dimen@<\eql@tagwidth@single@
    \let\eql@numbering@best@use\eql@true
  \fi
}
```

@numbering@best@eval **TODO:** describe

```
\def\eql@numbering@best@eval{%
  \ifdefined\eql@numbering@best@auto
    \ifdefined\eql@numbering@best@use\else
```

```
      \ifnum\eql@numbering@target@>\z@
        \eql@numbering@best@test
      \fi
    \fi
  \fi
  \ifdefined\eql@numbering@best@use
    \eql@numbering@best@find
  \fi
}
```

I.5 Tag Margin

TODO: describe **TODO:** if a tag margin is installed for a single line, it will shift the center even if there is no tag or importantly if a tag has been raised.

djust@calc@tagmargin

```
\def\eql@adjust@calc@tagmargin{%
  \ifdefined\eql@tagmargin@val
    \eql@tagmargin@\glueexpr\eql@tagmargin@val\relax
  \else
    \eql@tagmargin@\eql@tagwidth@max@
    \ifdim\eql@tagmargin@>\z@
      \advance\eql@tagmargin@-\eql@tagsepmin@
    \fi
  \fi

  \dimen@\eql@tagrows@\p@
  \ifnum\eql@totalrows@=\@ne
    \ifnum\eql@tagrows@=\@ne
      \advance\dimen@1sp\relax
    \fi
  \fi
  \ifdim\dimen@>\eql@totalrows@\eql@tagmargin@ratio@\else
    \eql@tagmargin@\z@
  \fi

  \@tempdima\dimexpr\displaywidth
      -\eql@totalwidth@-\eql@columns@inter@\eql@colsepmin@\relax
  \@tempdimb\dimexpr\@tempdima-\tw@\eql@tagmargin@\relax
  \ifdim\@tempdimb>\z@
    \ifdim\eql@tagmargin@threshold\@tempdima<\@tempdimb
      \eql@tagmargin@\z@
    \fi
  \fi
}
```

I.6 Single Column

ql@adjust@calc@lines

```
\def\eql@adjust@calc@lines{%
  \eql@totalcolumns@\@ne
  \eql@columns@inter@\z@
  \eql@colsep@\z@
  \ifdefined\eql@layoutleft
    \eql@marginleft@\glueexpr\eql@layoutleftmargin\relax
    \eql@marginleft@min@\glueexpr\eql@layoutleftmarginmin\relax
    \ifdim\eql@marginleft@<\eql@marginleft@min@
      \eql@marginleft@\eql@marginleft@min@
    \fi
    \dimen@\glueexpr\eql@layoutleftmarginmax\relax
    \ifdim\eql@marginleft@>\dimen@
      \eql@marginleft@\dimen@
    \fi
    \eql@marginright@\z@
    \eql@centeroffset@\z@
  \else
    \eql@adjust@calc@tagmargin
    \ifdefined\eql@paddingleft@val
      \eql@marginleft@\dimexpr
          (\displaywidth-\eql@totalwidth@-\eql@tagmargin@)/\tw@
          -\glueexpr\eql@paddingleft@val\relax\relax
      \ifdim\eql@marginleft@<\z@
        \eql@marginleft@\z@
      \fi
    \else
      \eql@marginleft@\z@
    \fi
    \ifdefined\eql@paddingright@val
      \eql@marginright@\dimexpr
          (\displaywidth-\eql@totalwidth@-\eql@tagmargin@)/\tw@
          -\glueexpr\eql@paddingright@val\relax\relax
      \ifdim\eql@marginright@<\z@
        \eql@marginright@\z@
      \fi
    \else
      \eql@marginright@\z@
    \fi
    \ifdim\eql@tagmargin@>\z@
      \ifdefined\eql@tagsleft
        \ifdim\eql@marginleft@<\eql@tagsepmin@
          \eql@marginleft@\eql@tagsepmin@
        \fi
        \advance\eql@marginleft@\eql@tagmargin@
        \advance\eql@centeroffset@\eql@tagmargin@
      \else
        \ifdim\eql@marginright@<\eql@tagsepmin@
          \eql@marginright@\eql@tagsepmin@
        \fi
        \advance\eql@marginright@\eql@tagmargin@
        \advance\eql@centeroffset@-\eql@tagmargin@
      \fi
    \fi
    \eql@marginleft@min@\z@
    \eql@centeroffset@\dimexpr\eql@marginright@-\eql@marginleft@
        \ifdefined\eql@tagsleft+\else-\fi\eql@tagmargin@\relax
  \fi

  \eql@totalwidth@\dimexpr\displaywidth
      -\eql@marginleft@-\eql@marginright@\relax
}
```

I.7 Multiple Columns

The following code computes the horizontal placement of columns. It distributes the columns evenly according to the layout presets and then determines whether there is enough space to place an equation tag on each line. If not, the intercolumn spacing and the space at the opposite margin can be reduced.

`@adjust@calc@columns` Main method to adjust column placement and spacing:

```
\def\eql@adjust@calc@columns{%
```

If there is just a single alignment stucture, there will be no intercolumn space that might stretch to adjust the columns to the margins. We disable fulllength to avoid a division by zero. Also guard against no columns at all (empty body), just in case:

```
  \ifnum\eql@totalcolumns@<\thr@@
    \eql@totalcolumns@\tw@
    \let\eql@columns@fulllength\eql@false
  \fi
```

Determine the number of intercolumn spaces `\eql@columns@inter@`:

```
  \eql@columns@inter@\numexpr(\eql@totalcolumns@-\tw@)/\tw@\relax
```

Evaluate the minimum intercolumn space which we will need often:

```
  \eql@colsepmin@\glueexpr\eql@colsepmin@val\relax
```

Determine the left or target margin width depending on the layout:

```
  \ifdefined\eql@layoutleft
    \eql@marginleft@\glueexpr\eql@layoutleftmargin\relax
    \eql@marginleft@min@\glueexpr\eql@layoutleftmarginmin\relax
    \ifdim\eql@marginleft@<\eql@marginleft@min@
      \eql@marginleft@\eql@marginleft@min@
    \fi
  \else
```

Get the desired tag margin, increase by minimum tag separation if columns are aligned to the margins. Cancel tag margin if too wide:

```
    \eql@adjust@calc@tagmargin
    \ifdefined\eql@columns@fulllength
      \ifdim\eql@tagmargin@>\z@
        \advance\eql@tagmargin@\eql@tagsepmin@
      \fi
    \fi
    \ifdim\eql@tagmargin@>\dimexpr\displaywidth-\eql@totalwidth@
        -\eql@columns@inter@\eql@colsepmin@\relax
      \eql@tagmargin@\z@
    \fi
    \eql@marginleft@min@\z@
  \fi
```

Compute the intercolumn space `\eql@colsep@`:

```
  \ifnum\eql@columns@inter@>\z@
```

Distribute the available horizontal space evenly onto the intercolumn spaces and the margins. Unless the columns are aligned to the margins, there are two margins in central alignment layout but only the right margin in left alignment layout:

```
    \eql@colsep@\dimexpr\displaywidth-\eql@totalwidth@\relax
    \ifdefined\eql@layoutleft
      \advance\eql@colsep@-\eql@marginleft@
    \else
      \advance\eql@colsep@-\eql@tagmargin@
    \fi
    \count@\eql@columns@inter@
    \ifdefined\eql@columns@fulllength\else
      \ifdefined\eql@layoutleft
        \advance\count@\@ne
      \else
        \advance\count@\tw@
      \fi
    \fi
    \divide\eql@colsep@\count@
```

Ensure that the intercolumn separation is within the specified bounds. Disable the upper bound if columns are to be aligned to the margins:

```
    \ifdim\eql@colsep@<\eql@colsepmin@
      \eql@colsep@\eql@colsepmin@
    \else
      \ifdefined\eql@columns@fulllength\else
        \dimen@\glueexpr\eql@colsepmax@val\relax
        \ifdim\eql@colsep@>\dimen@
          \eql@colsep@\dimen@
        \fi
      \fi
    \fi
  \else
```

For a single column, set the column separation to the minimum amount:

```
    \eql@colsep@\eql@colsepmin@
  \fi
```

Compute the left margin `\eql@marginleft@` depending on the layout:

```
  \ifdefined\eql@layoutleft
```

Set the default value:

```
    \ifdim\eql@colsep@=\eql@colsepmin@
```

If in left alignment layout the intercolumn space has been adjusted, compute the available space, determine left margin and make sure it is between the minimum and the default value:

```
      \dimen@\dimexpr\displaywidth-\eql@totalwidth@
          -\eql@columns@inter@\eql@colsep@\relax
      \ifdim\dimen@<\eql@marginleft@
        \ifdim\dimen@<\eql@marginleft@min@
          \eql@marginleft@\eql@marginleft@min@
        \else
          \eql@marginleft@\dimen@
        \fi
      \fi
    \fi
  \else
```

In central alignment mode with column aligned to the margins, set margin to zero:

```
    \ifdefined\eql@columns@fulllength
      \eql@marginleft@\z@
```

In central alignment mode with margins, distribute the available space equally to both margins, or remove the left margin if insufficient:

```
    \else
      \eql@marginleft@\dimexpr(\displaywidth-\eql@totalwidth@
          -\eql@columns@inter@\eql@colsep@-\eql@tagmargin@)/\tw@\relax
      \ifdim\eql@marginleft@<\z@
        \eql@marginleft@\z@
      \fi
    \fi
```

Add tag margin in case of left tags:

```
    \ifdefined\eql@tagsleft
      \advance\eql@marginleft@\eql@tagmargin@
    \fi
  \fi
```

Find the best row for tag placement:

```
  \eql@numbering@best@eval
```

Next consider all rows with tags and adjust the intercolumn and margin space to make the tags fit into the available space at the corresponding side as far as possible. First, select code depending on tag placement:

```
  \ifdefined\eql@tagsleft
    \let\eql@adjust@columns@test\eql@adjust@columns@test@tagsleft
  \else
    \let\eql@adjust@columns@test\eql@adjust@columns@test@tagsright
  \fi
```

Loop over all rows or select the single row containing the tag. Fetch the width data for the current row. If a tag is present, compute the available space and try to adjust spaces if needed:

```
  \ifnum\eql@numbering@target@<\z@
    \eql@widthdata@for{%
      \ifdim\eql@tagwidth@>\z@
        \eql@widthdata@calc
        \eql@adjust@columns@test
      \fi
    }%
  \else
    \ifnum\eql@numbering@target@>\z@
      \ifnum\eql@numbering@target@>\eql@totalrows@\else
        \eql@widthdata@get\eql@numbering@target@
        \eql@tagwidth@\eql@tagwidth@single@
        \eql@widthdata@calc
        \eql@adjust@columns@test
      \fi
    \fi
  \fi
```

From now on `\eql@totalwidth@` will include the left margin and the total intercolumn separation:

```
  \advance\eql@totalwidth@\dimexpr
      \eql@columns@inter@\eql@colsep@+\eql@marginleft@\relax
```

```
}
```

Placement for Right Tags.

lumns@test@tagsright Test whether the spacing can be adjusted to make the current row fit:

```
\def\eql@adjust@columns@test@tagsright{%
```

The register `\@tempdima` will hold the amount of available space. **TODO:** does this apply equally to left alignment layout?

```
  \@tempdima\dimexpr\displaywidth-\eql@line@width@-\eql@tagwidth@\relax
```

Test whether the space at the end of the row is sufficient to hold the tag with the current settings.

```
  \ifdim\@tempdima<\dimexpr
      \eql@marginleft@+\eql@line@widthsep@\eql@colsep@\relax
```

If not, determine whether the row and tag may at all fit into the available space with minimal intercolumn spaces and minimal left margin (in left alignment layout).

```
    \ifdim\@tempdima<\dimexpr
        \eql@marginleft@min@+\eql@line@widthsep@\eql@colsepmin@\relax\else
```

If so, hand over to `\eql@adjust@columns@modify@tagsright`.

```
      \eql@adjust@columns@modify@tagsright
    \fi
  \fi
}
```

mns@modify@tagsright Adjust the intercolumn space and left margin to make the row fit.

```
\def\eql@adjust@columns@modify@tagsright{%
```

If there are any intercolumn spaces that contribute to the available space, determine how much intercolumn separation would be needed while keeping the current left margin fixed (in left alignment layout). In central alignment layout, assume that the left margin will be adjusted to match the intercolumn separation by stepping the number of columns to divide by.

```
  \ifnum\eql@line@widthsep@>\z@
    \dimen@\@tempdima
    \count@\eql@line@widthsep@
    \ifdefined\eql@layoutleft
      \advance\dimen@-\eql@marginleft@
    \else
      \ifdefined\eql@columns@fulllength\else
        \advance\count@\@ne
      \fi
    \fi
    \divide\dimen@\count@
```

If smaller, reduce the intercolumn separation, but make sure to not exceed the minumum allowed value.

```
    \ifdim\dimen@<\eql@colsep@
      \ifdim\dimen@<\eql@colsepmin@
        \eql@colsep@\eql@colsepmin@
      \else
```

```
        \eql@colsep@\dimen@
      \fi
    \fi
  \fi
```

Now adjust the left margin as much as needed to fit the contents.

```
  \dimen@\dimexpr\@tempdima-\eql@line@widthsep@\eql@colsep@\relax
  \ifdim\eql@marginleft@>\dimen@
    \eql@marginleft@\dimen@
  \fi
}
```

Placement for Left Tags.

olumns@test@tagsleft Test whether the spacing can be adjusted to make the current row fit:

```
\def\eql@adjust@columns@test@tagsleft{%
```

The register `\@tempdima` will hold the deficit amount of space at the beginning of the row without adjustable space, and the register `\count@` will hold the number of intercolumn spaces that would contribute to space adjustments.

```
  \count@\numexpr\eql@columns@inter@-\eql@line@availsep@\relax
  \@tempdima\dimexpr\eql@tagwidth@-\eql@line@avail@\relax
```

Test whether the space at the beginning of the row is sufficient to hold the tag with the current settings.

```
  \ifdim\@tempdima>\dimexpr
      \eql@marginleft@+\eql@line@availsep@\eql@colsep@\relax
```

If not, first verify that the tag will fit the line (or the maxumal left margin in left alignment layout).

```
    \ifdim\eql@tagwidth@<
        \ifdefined\eql@layoutleft
          \glueexpr\eql@layoutleftmarginmax\relax
        \else
          \displaywidth
        \fi
```

If so, determine whether the row and tag may at all fit into the available space with minimal intercolumn spaces.

```
      \ifdim\@tempdima>\dimexpr
          \displaywidth-\eql@totalwidth@-\count@\eql@colsepmin@\relax\else
```

If so, hand over to `\eql@adjust@columns@modify@tagsleft`.

```
        \eql@adjust@columns@modify@tagsleft
      \fi
    \fi
  \fi
}
```

umns@modify@tagsleft Adjust the intercolumn space and left margin to make the row fit.

```
\def\eql@adjust@columns@modify@tagsleft{%
```

If there are any intercolumn spaces that contribute to the available space, determine how much intercolumn separation would be needed while keeping the current right margin fixed. In central alignment layout, assume that the right margin will be adjusted to match the intercolumn separation by stepping the number of columns to divide by.

```
  \ifnum\count@>\z@
    \dimen@\dimexpr\displaywidth-\eql@totalwidth@-\@tempdima\relax
    \ifdefined\eql@columns@fulllength\else
      \advance\count@\@ne
    \fi
    \divide\dimen@\count@
```

If smaller, reduce the intercolumn separation, but make sure to not exceed the minumum allowed value. Also adjust the left margin to keep the right margin fixed.

```
    \ifdim\dimen@<\eql@colsep@
      \ifdim\dimen@<\eql@colsepmin@
        \dimen@\eql@colsepmin@
      \fi
      \advance\dimen@-\eql@colsep@
      \advance\eql@marginleft@-\eql@columns@inter@\dimen@
      \advance\eql@colsep@\dimen@
    \fi
  \fi
```

Now adjust the left margin as much as needed to fit the contents.

```
  \dimen@\dimexpr\@tempdima-\eql@line@availsep@\eql@colsep@\relax
  \ifdim\eql@marginleft@<\dimen@
    \eql@marginleft@\dimen@
  \fi
}
```

J Single Column Arrangement

The following code adjusts individual lines of equations for the equation and lines mode according to the selected layout and shape.

J.1 Supporting Definitions

\inf@bad The \inf@bad constant is for testing overfull boxes:

```
\ifdefined\inf@bad\else%
  \newcount\inf@bad
  \inf@bad1000000\relax
\fi
```

\eql@restore@hfuzz \eql@save@hfuzz We need to change the value of \hfuzz temporarily. The method \eql@save@hfuzz stores the value for recovery through \eql@restore@hfuzz:

```
\let\eql@restore@hfuzz\@empty
\def\eql@save@hfuzz{\edef\eql@restore@hfuzz{\hfuzz\the\hfuzz\relax}}
```

\eql@alignbadness@ \eql@tagbadness@ The registers \eql@alignbadness@ and \eql@tagbadness@ store the allowable badness threshold for shrinking equation lines to the intended margin or to fit into the line at all before the tag is raised or lowered:

```
\newcount\eql@alignbadness@
\newcount\eql@tagbadness@
```

```
\newcount\eql@arrange@badness@
\eql@alignbadness@\inf@bad
\eql@tagbadness@\inf@bad
```

J.2 Arrangement Methods

`\eql@arrange@try` Try to fit the current equation line in the available space. Argument `#1` specifies the amount of reserved space. Unpack the box `\eql@cellbox@`, replace the previous kerning with the new reserved space, and save the box back into `\eql@cellbox@`:

```
\def\eql@arrange@try#1{%
  \ifdim#1>\dimexpr\displaywidth-\eql@cellwidth@\relax
    \setbox\eql@cellbox@\hbox to\displaywidth{%
      \unhbox\eql@cellbox@\unkern\kern#1}%
    \eql@arrange@badness@\badness
  \else
    \eql@arrange@badness@\m@ne
  \fi
}
```

`\eql@arrange@print` We have found the final adjustment of the current line, so we typeset it with initial and final space adjustments `#1` and `#2`, respectively. Restore the original value for `\hfuzz`:

```
\def\eql@arrange@print#1#2{%
  \eql@restore@hfuzz
  \hbox to\displaywidth{%
    #1%
    \unhbox\eql@cellbox@\unkern
    #2%
    \eql@tagging@mathaddlast
  }%
}
```

`ange@print@alignleft`
`ge@print@aligncenter`
`nge@print@alignright`
Fit the current equation line with the selected alignment within a given left and right margins `#1` and `#2`. If we're on the first line, adjust `\eql@display@firstavail@` to the mininum left available space we can guarantee:

```
\def\eql@arrange@print@alignleft#1#2{%
  \ifnum\eql@row@=\@ne
    \global\eql@display@firstavail@#1%
  \fi
  \eql@arrange@print{\kern#1}{\kern#2}%
}

\def\eql@arrange@print@alignright#1#2{%
  \ifnum\eql@row@=\@ne
    \global\eql@display@firstavail@\dimexpr
        \displaywidth-\eql@cellwidth@-#2\relax
  \fi
  \eql@arrange@print{\kern#1\hfil}{\unskip\kern#2}%
}

\def\eql@arrange@print@aligncenter#1{%
  \ifnum\eql@row@=\@ne
    \global\eql@display@firstavail@\dimexpr
        (\displaywidth-\eql@cellwidth@+#1)/2\relax
  \fi
  \ifdim#1>\z@
```

```
    \eql@arrange@print{\kern#1\hfil}{}%
  \else
    \eql@arrange@print{\hfil}{\kern-#1}%
  \fi
}
```

\eql@arrange@init Initialise the horizontal adjustment framework. Turn off overfull box messages temporarily – otherwise there would be unwanted extra ones emitted during our measuring operations. Select the shape scheme:

```
\def\eql@arrange@init{%
  \eql@save@hfuzz
  \hfuzz\maxdimen
  \eql@shape@select
}
```

ql@arrange@print@tag @arrange@print@notag Select the appropriate adjustment method depending on the current alignment position, the selected tag placement if any:

```
\def\eql@arrange@print@tag{%
  \eql@tagging@tagaddbox
  \csname eql@arrange%
    @\ifcase\eql@shape@pos@ alignleft\or aligncenter\or alignright\fi
    @init\endcsname
  \csname eql@arrange%
    @\ifcase\eql@shape@pos@ alignleft\or aligncenter\or alignright\fi
    @\ifdefined\eql@tagsleft tagsleft\else tagsright\fi
\endcsname
}
```

```
\def\eql@arrange@print@notag{%
  \eql@tagging@tagaddbox
  \csname eql@arrange%
    @\ifcase\eql@shape@pos@ alignleft\or aligncenter\or alignright\fi
    @init\endcsname
  \csname eql@arrange%
    @\ifcase\eql@shape@pos@ alignleft\or aligncenter\or alignright\fi
    @notag\endcsname
}
```

J.3 Central Alignment

TODO: describe

```
\def\eql@arrange@aligncenter@init{%
  \eql@tagging@aligncenter
  \eql@line@offset@\dimexpr\tw@\eql@shape@amount@
       +\eql@marginleft@-\eql@marginright@+\eql@centeroffset@\relax
}
```

TODO: describe

```
\def\eql@arrange@aligncenter@notag{%
  \ifdim\dimexpr\displaywidth-\eql@cellwidth@\relax>
      \ifdim\eql@line@offset@<\eql@marginleft@min@
        \dimexpr\tw@\eql@marginleft@min@-\eql@line@offset@\relax
      \else
        \eql@line@offset@
      \fi
```

```
    \eql@arrange@print@aligncenter\eql@line@offset@
  \else
    \ifdim\eql@line@offset@<\eql@marginleft@min@
      \eql@arrange@print@alignleft\eql@marginleft@min@\z@
    \else
      \eql@arrange@print@alignright\eql@marginleft@min@\z@
    \fi
  \fi
}
```

TODO: describe

```
\def\eql@arrange@aligncenter@tagsright{%
  \ifdim\dimexpr\displaywidth-\eql@cellwidth@\relax>
      \ifdim\eql@line@offset@<\dimexpr\eql@marginleft@min@-\eql@tagwidth@\relax
        \dimexpr\tw@\eql@marginleft@min@-\eql@line@offset@\relax
      \else
        \dimexpr\tw@\eql@tagwidth@+\eql@line@offset@\relax
      \fi
    \eql@arrange@print@aligncenter\eql@line@offset@
    \eql@tagbox@print@tagsright
  \else
    \eql@arrange@try{\dimexpr\eql@tagwidth@+\eql@marginleft@min@\relax}%
    \ifnum\eql@arrange@badness@<\eql@tagbadness@
      \ifdim\eql@line@offset@<\dimexpr\eql@marginleft@min@-\eql@tagwidth@\relax
        \eql@arrange@print@alignleft\eql@marginleft@min@\eql@tagwidth@
      \else
        \eql@arrange@print@alignright\eql@marginleft@min@\eql@tagwidth@
      \fi
      \eql@tagbox@print@tagsright
    \else
      \eql@arrange@aligncenter@notag
      \eql@tagbox@print@tagsright@raise
    \fi
  \fi
}

\def\eql@arrange@aligncenter@tagsleft{%
  \ifdim\eql@tagwidth@>\eql@marginleft@min@
    \ifdim\dimexpr\displaywidth-\eql@cellwidth@\relax>
        \ifdim\eql@line@offset@<\eql@tagwidth@
          \dimexpr\tw@\eql@tagwidth@-\eql@line@offset@\relax
        \else
          \eql@line@offset@
        \fi
      \eql@tagbox@print@tagsleft
      \eql@arrange@print@aligncenter\eql@line@offset@
    \else
      \eql@arrange@try\eql@tagwidth@
      \ifnum\eql@arrange@badness@<\eql@tagbadness@
        \eql@tagbox@print@tagsleft
        \ifdim\eql@line@offset@<\eql@tagwidth@
          \eql@arrange@print@alignleft\eql@tagwidth@\z@
        \else
          \eql@arrange@print@alignright\eql@tagwidth@\z@
        \fi
      \else
        \eql@tagbox@print@tagsleft@raise
        \eql@arrange@aligncenter@notag
      \fi
    \fi
  \else
    \eql@tagbox@print@tagsleft
    \eql@arrange@aligncenter@notag
  \fi
  \eql@display@firstavail@set\z@
}
```

J.4 Left Alignment

```
\def\eql@arrange@alignleft@init{%
  \eql@tagging@alignleft
  \eql@line@offset@\dimexpr\eql@marginleft@+\eql@shape@amount@\relax
  \ifdim\eql@line@offset@<\eql@marginleft@min@
    \eql@line@offset@\eql@marginleft@min@
  \fi
}
```

```
\def\eql@arrange@alignleft@notag{%
  \ifdim\eql@line@offset@>\eql@marginleft@min@
    \eql@arrange@try\eql@line@offset@
    \ifnum\eql@arrange@badness@<\eql@alignbadness@
      \eql@arrange@print@alignleft\eql@line@offset@\z@
    \else
      \eql@arrange@print@alignright\eql@marginleft@min@\z@
    \fi
  \else
    \eql@arrange@print@alignleft\eql@marginleft@min@\z@
  \fi
}
```

```
\def\eql@arrange@alignleft@tagsright{%
  \eql@arrange@try{\dimexpr\eql@line@offset@+\eql@tagwidth@\relax}%
  \ifnum\eql@arrange@badness@<\eql@alignbadness@
    \eql@arrange@print@alignleft\eql@line@offset@\eql@tagwidth@
    \eql@tagbox@print@tagsright
  \else
    \ifdim\eql@line@offset@>\eql@marginleft@min@
      \eql@arrange@try{\dimexpr\eql@marginleft@min@+\eql@tagwidth@\relax}%
    \fi
    \ifnum\eql@arrange@badness@<\eql@tagbadness@
      \eql@arrange@print@alignright\eql@marginleft@min@\eql@tagwidth@
      \eql@tagbox@print@tagsright
    \else
      \eql@arrange@alignleft@notag
      \eql@tagbox@print@tagsright@raise
    \fi
  \fi
}
```

```
\def\eql@arrange@alignleft@tagsleft{%
  \ifdim\eql@tagwidth@>\eql@marginleft@min@
    \ifdim\eql@line@offset@>\eql@tagwidth@
      \eql@arrange@try\eql@line@offset@
      \ifnum\eql@arrange@badness@<\eql@alignbadness@
        \eql@tagbox@print@tagsleft
        \eql@arrange@print@alignleft\eql@line@offset@\z@
      \else
        \eql@arrange@try\eql@tagwidth@
      \ifnum\eql@arrange@badness@<\eql@tagbadness@
        \eql@tagbox@print@tagsleft
        \eql@arrange@print@alignright\eql@tagwidth@\z@
      \else
        \eql@tagbox@print@tagsleft@raise
        \eql@arrange@print@alignright\eql@marginleft@min@\z@
      \fi
    \fi
  \else
    \eql@arrange@try\eql@tagwidth@
    \ifnum\eql@arrange@badness@<\eql@tagbadness@
      \eql@tagbox@print@tagsleft
      \eql@arrange@print@alignleft\eql@tagwidth@\z@
    \else
      \eql@tagbox@print@tagsleft@raise
      \eql@arrange@alignleft@notag
    \fi
  \fi
\else
  \eql@tagbox@print@tagsleft
  \eql@arrange@alignleft@notag
\fi
\eql@display@firstavail@set\z@
}
```

J.5 Right Alignment

```
\def\eql@arrange@alignright@init{%
  \eql@tagging@alignright
  \eql@line@offset@\dimexpr\eql@marginright@-\eql@shape@amount@\relax
  \ifdim\eql@line@offset@<\z@
    \eql@line@offset@\z@
  \fi
}
```

TODO: describe

```
\def\eql@arrange@alignright@notag{%
  \ifdim\eql@line@offset@>\z@
    \eql@arrange@try{\dimexpr\eql@marginleft@min@+\eql@line@offset@\relax}%
    \ifnum\eql@arrange@badness@<\eql@alignbadness@
      \eql@arrange@print@alignright\eql@marginleft@min@\eql@line@offset@
    \else
      \eql@arrange@print@alignleft\eql@marginleft@min@\z@
    \fi
  \else
    \eql@arrange@print@alignright\eql@marginleft@min@\z@
  \fi
}
```

TODO: describe

```
\def\eql@arrange@alignright@tagsright{%
  \ifdim\eql@line@offset@>\eql@tagwidth@
    \eql@arrange@try{\dimexpr\eql@marginleft@min@+\eql@line@offset@\relax}%
    \ifnum\eql@arrange@badness@<\eql@alignbadness@
      \eql@arrange@print@alignright\eql@marginleft@min@\eql@line@offset@
      \eql@tagbox@print@tagsright
    \else
      \eql@arrange@try{\dimexpr\eql@marginleft@min@+\eql@tagwidth@\relax}%
      \ifnum\eql@arrange@badness@<\eql@tagbadness@
```

```
        \eql@arrange@print@alignleft\eql@marginleft@min@\eql@tagwidth@
        \eql@tagbox@print@tagsright
      \else
        \eql@arrange@print@alignleft\eql@marginleft@min@\z@
        \eql@tagbox@print@tagsright@raise
      \fi
    \fi
  \else
    \eql@arrange@try{\dimexpr\eql@marginleft@min@+\eql@tagwidth@\relax}%
    \ifnum\eql@arrange@badness@<\eql@tagbadness@
      \eql@arrange@print@alignright\eql@marginleft@min@\eql@tagwidth@
      \eql@tagbox@print@tagsright
    \else
      \eql@arrange@alignright@notag
      \eql@tagbox@print@tagsright@raise
    \fi
  \fi
}
```

TODO: describe

```
\def\eql@arrange@alignright@tagsleft{%
  \ifdim\eql@tagwidth@>\eql@marginleft@min@
    \eql@arrange@try{\dimexpr\eql@line@offset@+\eql@tagwidth@\relax}%
    \ifnum\eql@arrange@badness@<\eql@alignbadness@
      \eql@tagbox@print@tagsleft
      \eql@arrange@print@alignright\eql@tagwidth@\eql@line@offset@
    \else
      \ifdim\eql@line@offset@>\z@
        \eql@arrange@try\eql@tagwidth@
      \fi
      \ifnum\eql@arrange@badness@<\eql@tagbadness@
        \eql@tagbox@print@tagsleft
        \eql@arrange@print@alignleft\eql@tagwidth@\z@
      \else
        \eql@tagbox@print@tagsleft@raise
        \eql@arrange@alignright@notag
      \fi
    \fi
  \else
    \eql@tagbox@print@tagsleft
    \eql@arrange@alignright@notag
  \fi
  \eql@display@firstavail@set\z@
}
```

K Equations Box Environment

TODO: describe

TODO: fixed width version (works only towards intercolumn stretch)?

TODO: vspace?!

K.1 Line Breaks

\eql@box@cr

```
\protected\def\eql@box@cr{%
```

```
  \eql@ampprotecttwo{\eql@ifnextchar@tight[}\eql@box@cr@skip\eql@box@cr@
}
\def\eql@box@cr@{%
  \eql@punct@apply@line
  \eql@hook@lineout
  \eql@box@lastcell
  \cr
}
\def\eql@box@cr@skip[#1]{%
  \eql@box@cr@
  \noalign{%
    \vskip\glueexpr#1\relax
  }%
}
```

K.2 Stacked Mode

```
\def\eql@box@lastcell@stacked{&\omit\kern-2\eql@colsep@}
```

```
\def\eql@box@open@stacked{%
% \TODO templates
  \eql@shape@align@enable
  \let\eql@box@lastcell\eql@box@lastcell@stacked
  \everycr{\noalign{%
⟨dev⟩\eql@dev{starting line \the\eql@row@}%
    \global\advance\eql@row@\@ne
  }}%
  \tabskip\z@skip
  \halign\bgroup
    &%
      \eql@shape@pos@\m@ne
      \setbox\eql@cellbox@\hbox{%
        \eql@strut@cell
        \@lign
        $\m@th\displaystyle
          \eql@hook@colin
          ##%
          \eql@punct@apply@col
          \eql@hook@colout
          \eql@tagging@mathsave
        $%
        \eql@tagging@mathaddlast
      }%
      \ifdefined\eql@frame@cmd
        \eql@frame@print
      \fi
      \ifnum\eql@shape@pos@<\z@
        \ifcase-\eql@shape@pos@
        \or
          \eql@shape@eval
        \or
          \@tempdima\eql@shape@amount@
          \eql@shape@eval
          \eql@shape@amount@\@tempdima
        \or
          \@tempdima\eql@shape@amount@
          \eql@shape@eval
          \advance\eql@shape@amount@\@tempdima
        \fi
      \fi
      \ifcase\eql@shape@pos@
        \kern\eql@shape@amount@
        \box\eql@cellbox@
        \hskip\glueexpr\eql@paddingleft@+\eql@paddingright@
          -\eql@shape@amount@+\@flushglue\relax
        \eql@tagging@alignleft
      \or
        \hskip\glueexpr\eql@paddingleft@+\eql@shape@amount@+\@flushglue\relax
        \box\eql@cellbox@
        \hskip\glueexpr\eql@paddingright@-\eql@shape@amount@+\@flushglue\relax
        \eql@tagging@aligncenter
      \or
        \hskip\glueexpr\eql@paddingleft@+\eql@paddingright@
          +\eql@shape@amount@+\@flushglue\relax
        \box\eql@cellbox@
        \kern-\eql@shape@amount@
        \eql@tagging@alignright
      \fi
      \tabskip\eql@colsep@\relax
    \crcr
    \noalign{%
      \eql@hook@blockbefore
    }%
    \eql@hook@blockin
}
\def\eql@mode@stacked{\let\eql@box@open\eql@box@open@stacked}
```

K.3 Aligned Mode

```
\def\eql@box@lastcell@odd{%
  &\omit
  \ifdefined\eql@frame@cmd
    \eql@frame@measure
    \advance\eql@prevwidth@\eql@frame@margin@
    \eql@frame@print
  \fi
  \kern-\eql@prevwidth@
  \unhbox\eql@cellbox@
  \hfil
  &\omit\kern-\eql@colsep@
}%
\def\eql@box@lastcell@even{&\omit\kern-\eql@colsep@}

\def\eql@box@open@aligned{%
% \TODO templates
  \eql@shape@align@disable
  \let\eql@box@lastcell\@empty
  \everycr{\noalign{%
⟨dev⟩\eql@dev{starting new line}%
  }}%
  \tabskip\z@skip
  \halign\bgroup
    &%
      \let\eql@box@lastcell\eql@box@lastcell@odd
      \global\setbox\eql@cellbox@\hbox{%
        \eql@strut@cell
        \@lign
        $\m@th\displaystyle
```

```
          \eql@hook@colin
          ##%
          \eql@class@innerleft
          \eql@hook@innerleft
          \eql@tagging@mathsave
        $%
        \eql@tagging@mathaddlast
      }%
      \global\eql@prevwidth@\wd\eql@cellbox@
      \hfil
      \kern\wd\eql@cellbox@
      \ifdefined\eql@frame@cmd
        \eql@frame@measure
        \kern\eql@frame@margin@
      \fi
      \tabskip\z@skip
    &%
      \let\eql@box@lastcell\eql@box@lastcell@even
      \setbox\eql@cellbox@\hbox{%
        \unhbox\eql@cellbox@
        \eql@strut@cell
        \@lign
        $\m@th\displaystyle
          \eql@hook@innerright
          \eql@class@innerright@sel
          ##%
          \eql@punct@apply@col
          \eql@hook@colout
          \eql@tagging@mathsave
        $%
        \eql@tagging@mathaddlast
      }%
      \ifdefined\eql@frame@cmd
        \eql@frame@measure
        \advance\eql@prevwidth@\eql@frame@margin@
        \eql@frame@print
      \fi
      \kern-\eql@prevwidth@
      \unhbox\eql@cellbox@
      \hfil
      \tabskip\eql@colsep@\relax
    \crcr
    \noalign{%
      \eql@hook@blockbefore
    }%
    \eql@hook@blockin
}
```

```
\def\eql@mode@aligned{\let\eql@box@open\eql@box@open@aligned}
```

K.4 Main

```
\let\eql@box@box\vcenter
\let\eql@box@open\@undefined
\let\eql@box@frame\@firstofone
\def\eql@box@wrap#1#2{\def\eql@box@frame##1{#1##1#2}}
```

```
\def\eql@box@close{%
    \ifvmode\else
      \global\eql@totalrows@\eql@row@
      \eql@punct@apply@block
      \eql@box@cr@
    \fi
    \noalign{%
      \eql@hook@blockafter
    }%
    \eql@tagging@tablesaveinner
  \egroup
}
```

\eql@box@vcenter

```
\def\eql@box@vcenter#1{%
  \ifmmode
    \vcenter{#1}%
  \else
    $\m@th\vcenter{#1}$%
  \fi
}
```

\eql@box@start

```
\let\eql@box@endmath\eql@false
\def\eql@box@start{%
  \relax
  \ifmmode
    \let\eql@box@endmath\eql@false
  \else
    \let\eql@box@endmath\eql@true
    \expandafter$%$
  \fi
  \eql@box@processopt
  \eql@stack@save@boxed
  \let\eql@layoutleft\eql@false
  \eql@row@\z@
  \eql@totalrows@\@M
  \eql@shape@select
  \setbox\z@\ifx\eql@box@box\vcenter
    \expandafter\vbox
  \else
    \expandafter\eql@box@box
  \fi\bgroup
    \eql@display@nest
    \let\\\eql@box@cr
    \eql@spread@set
    \eql@strut@make
    \eql@box@open
}
```

\eql@box@end

```
\def\eql@box@end{%
    \eql@box@close
  \egroup
  \eql@box@frame{%
    \ifdefined\eql@display@marginleft
      \hskip\glueexpr\eql@display@marginleft\relax
    \fi
    \ifx\eql@box@box\vcenter
```

```
      \eql@box@vcenter{\unvbox\z@}%
    \else
      \box\z@
    \fi
    \eql@tagging@tableaddinner
    \ifdefined\eql@display@marginright
      \hskip\glueexpr\eql@display@marginright\relax
    \fi
  }%
  \eql@stack@restore
  \ifdefined\eql@box@endmath
    \expandafter$%$
  \fi
}
```

K.5 Environment

equationsbox (*env.*)

```
\newenvironment{equationsbox}{%
⟨dev⟩\eql@dev@enterenv
  \eql@ampprotect\eql@box@testall\eql@box@start
}{%
  \eql@box@end
⟨dev⟩\eql@dev@leaveenv
}
```

```
\def\eql@box@testall{\eql@box@testtilde}
\def\eql@box@testtilde#1{%
  \eql@ifnextgobble@tight~%
    {\eqnaddopt{lines}\eql@box@testopt{#1}}%
    {\eql@box@testopt{#1}}}
\def\eql@box@testopt#1{%
  \eql@ifnextchar@tight[%]
    {\eql@box@addopt{#1}}%
    {#1}}
\def\eql@box@addopt#1[#2]{\eqnaddopt{#2}#1}
```

\eql@box@processopt **TODO:** describe

```
\def\eql@box@processopt{%
  \eql@frame@reset
  \let\eql@box@frame\@firstofone
  \let\eql@display@marginleft\@undefined
  \let\eql@display@marginright\@undefined
  \eql@nextopt@process{equationsbox}%
  \let\eql@punct@block\eql@punct@main
  \let\eql@punct@main\relax
  \eql@colsep@\glueexpr\eql@box@colsep\relax
  \ifdefined\eql@paddingleft@val
    \eql@paddingleft@\glueexpr\eql@paddingleft@val\relax
  \else
    \eql@paddingleft@\z@
  \fi
  \ifdefined\eql@paddingright@val
    \eql@paddingright@\glueexpr\eql@paddingright@val\relax
  \else
    \eql@paddingright@\z@
  \fi
  \eql@indent@\glueexpr\eql@indent@val\relax
}
```

L Single-Line Equation

TODO: describe

L.1 Native Mode

```
\def\eql@single@start@native{%
  \eql@display@init
  \eql@display@print
  \let\raisetag\eql@raisetag@default
  \eql@shape@align@disable
  \eql@hook@eqin
% \mathopen{}%
}%
```

TODO: describe

```
\def\eql@single@end@native{%
% \mathclose{}%
  \if@eqnsw
    \ifdefined\eql@tagsleft
      \leqno
    \else
      \eqno
    \fi
    \eql@compose@print
  \fi
  \eql@display@penalty
  \eql@display@vspace@native
}%
```

L.2 Print

```
\def\eql@single@start@print{%
  \eql@display@init
  \eql@display@print
  \eql@shape@align@enable

  \eql@totalrows@\@ne
  \eql@row@\@ne
  \eql@arrange@init
  \eql@shape@eval

  \prevgraf\numexpr\prevgraf+\@ne\relax
  \setbox\eql@cellbox@\hbox\bgroup
    \eql@restore@hfuzz
    \eql@strut@cell
    $\m@th\displaystyle%$
      \eql@hook@eqin
}

\def\eql@single@end@print{%
    \eql@tagging@mathsave
    $%$
    \hfil
    \kern\z@
  \egroup
  \prevgraf\numexpr\prevgraf-\@ne\relax
  \ifdefined\eql@frame@cmd
    \eql@frame@adjust
  \fi
  \eql@cellwidth@\wd\eql@cellbox@
  \eql@line@height@\ht\eql@cellbox@
  \eql@line@depth@\dp\eql@cellbox@
  \eql@totalwidth@\eql@cellwidth@
  \if@eqnsw
    \eql@tagbox@make\eql@compose@print
    \eql@tagrows@\@ne
  \else
    \eql@tagwidth@\z@
    \eql@tagrows@\z@
  \fi
  \eql@tagwidth@max@\eql@tagwidth@
  \eql@columns@inter@\z@
  \eql@adjust@calc@lines
  \eql@display@halign@init{}%
  \halign{##\crcr
    \noalign{\eql@display@halign@start}%
    \if@eqnsw
      \eql@arrange@print@tag
    \else
      \eql@arrange@print@notag
    \fi
    \cr
    \noalign{\eql@display@halign@end}%
    \eql@tagging@tablesavelines
  }%
  \eql@display@close
}
```

M Multi-Line with Single Column

M.1 Measure

TODO: describe

```
\def\eql@lines@measure@line@begin{%
⟨dev⟩\eql@dev{starting line \the\eql@row@}%
  \eql@numbering@measure@line@begin
  \eql@hook@linein
}
```

TODO: describe

```
\def\eql@lines@measure@line@end{%
  \eql@punct@apply@line
  \eql@hook@lineout
}
```

TODO: describe **TODO:** it would be an option to add the absolute shove amount to the calculation of the maximum width

```
\def\eql@lines@measure@cell{%
  \ifdefined\eql@frame@cmd
    \eql@frame@print
  \fi
  \eql@cellwidth@\wd\eql@cellbox@
  \eql@widthdata@startrow
  \eql@widthdata@savecell
  \kern\eql@cellwidth@
}
```

TODO: describe

```
\def\eql@lines@measure@tag{%
  \eql@tagwidth@\z@
  \ifnum\eql@numbering@target@<\z@
    \if@eqnsw
      \eql@tagbox@make\eql@compose@measure
    \fi
  \fi
  \eql@widthdata@savetag
}
```

`\eql@lines@measure`

```
\def\eql@lines@measure{%
⟨dev⟩\eql@dev@enter\eql@lines@measure
  \eql@measure@init\eql@lines@measure@line@begin\eql@lines@measure@line@end
  \eql@totalrows@\@M
  \eql@shape@select

  \setbox\z@\vbox{\halign{%
      \eql@shape@eval
      \setbox\eql@cellbox@\hbox{%
        \@lign
        $\m@th\displaystyle
          \eql@hook@colin
          ##%
          \eql@punct@apply@col
          \eql@hook@colout
        $%
      }%
      \eql@lines@measure@cell
      \eql@lines@measure@tag
    \crcr

    \noalign{%
      \eql@hook@blockbefore
    }%
    \eql@hook@blockin
    \eql@scan@body
    \ifvmode\else
      \eql@totalrows@\eql@row@
      \eql@punct@apply@block
      \eql@hook@blockout
      \eql@display@endline
      \cr
    \fi
    \omit
    \cr
    \noalign{%
```

```
      \eql@hook@blockafter
    }%
  }}%
```

```
  \eql@measure@close
```

```
  \setbox\z@\vbox{%
    \unvbox\z@
    \unpenalty
    \global\setbox\@ne\lastbox
  }%
  \eql@totalwidth@\wd\@ne
```

```
⟨dev⟩\eql@dev@leave\eql@lines@measure
}
```

M.2 Column Placement

TODO: describe Find the best row for tag placement:

```
\def\eql@lines@adjust{%
  \eql@adjust@calc@lines
  \eql@numbering@best@eval
}
```

M.3 Print

TODO: describe

nes@print@line@begin

```
\def\eql@lines@print@line@begin{%
⟨dev⟩\eql@dev{starting line \the\eql@row@}%
  \eql@numbering@print@line@begin
  \eql@hook@linein
}
```

TODO: describe

```
\def\eql@lines@print@line@end{%
  \eql@punct@apply@line
  \eql@hook@lineout
}
```

TODO: describe

```
\def\eql@lines@print@line@adjust{%
  \ifdefined\eql@frame@cmd
    \eql@frame@adjust
  \fi
  \eql@numbering@print@line@eval
  \eql@cellwidth@\wd\eql@cellbox@
  \eql@line@height@\ht\eql@cellbox@
  \eql@line@depth@\dp\eql@cellbox@
  \if@eqnsw
    \eql@tagbox@make\eql@compose@print
    \eql@arrange@print@tag
  \else
    \eql@arrange@print@notag
  \fi
}
```

TODO: describe

```
\def\eql@lines@print{%
⟨dev⟩\eql@dev@enter\eql@lines@print
  \eql@arrange@init
  \eql@display@halign@init\eql@lines@print@line@begin
  \eql@display@halign@letcr\eql@lines@print@line@end
  \tabskip\z@skip

  \halign{%
      \eql@shape@eval
      \setbox\eql@cellbox@\hbox{%
        \eql@restore@hfuzz
        \eql@strut@cell
        \@lign
        $\m@th\displaystyle
          \eql@hook@colin
          ##%
          \eql@punct@apply@col
          \eql@hook@colout
          \eql@tagging@mathsave
        $%
        \hfil
        \kern\z@
      }%
      \eql@lines@print@line@adjust
    \crcr

    \noalign{%
      \eql@display@halign@start
      \eql@numbering@print@block@begin
      \eql@hook@blockbefore
    }%
    \eql@hook@blockin
    \eql@scan@body
    \ifvmode\else
      \relax
      \eql@punct@apply@block
      \eql@hook@blockout
      \eql@display@endline
      \cr
    \fi
    \noalign{%
      \eql@hook@blockafter
      \eql@display@halign@end
⟨dev⟩\eql@dev@leave\eql@lines@print
    }%
    \eql@tagging@tablesavelines
  }%
}
```

N Multi-Line with Multiple Columns

TODO: describe

N.1 Support

TODO: describe

\eql@columns@add@amp
@columns@completerow

```
\def\eql@columns@add@amp#1{\if m#1&\omit\expandafter\eql@columns@add@amp\fi}
\def\eql@columns@completerow{%
  \count@\eql@totalcolumns@
  \advance\count@\@ne
  \advance\count@-\eql@column@
  \edef\eql@tmp{%
    \expandafter\eql@columns@add@amp\romannumeral\number\count@ 000q}%
  \eql@tmp
}
```

```
\def\eql@columns@overfull{%
  \dimen@\eql@line@width@
  \advance\dimen@-\hfuzz
  \ifdim\dimen@>\displaywidth
    \setbox\z@\hbox to\displaywidth{\hbox to\eql@line@width@{\hfil}}%
    \wd\z@\z@
    \ht\z@\eql@line@height@
    \dp\z@\eql@line@depth@
    \box\z@
  \fi
}
```

N.2 Measure

TODO: describe **TODO:** this is called also for extra line and concluding cr

s@measure@line@begin

```
\def\eql@columns@measure@line@begin{%
⟨dev⟩\eql@dev{starting line \the\eql@row@}%
  \global\eql@column@\z@
  \eql@numbering@measure@line@begin
  \eql@hook@linein
}
```

```
\def\eql@columns@measure@cell{%
  \eql@cellwidth@\wd\eql@cellbox@
  \ifdefined\eql@frame@cmd
    \eql@frame@measure
    \advance\eql@cellwidth@\eql@frame@margin@
    \ifodd\eql@column@\else
      \eql@frame@reset
    \fi
  \fi
  \ifnum\eql@column@=\@ne
    \eql@widthdata@startrow
  \fi
  \ifodd\eql@column@
    \eql@shape@pos@\tw@
  \else
    \eql@shape@pos@\z@
  \fi
  \eql@shape@amount@\z@
  \eql@widthdata@savecell
  \ifodd\eql@column@\else
    \eql@widthdata@savesep
  \fi
  \kern\eql@cellwidth@
}
```

mns@measure@line@end

```
\def\eql@columns@measure@line@end{%
  \eql@punct@apply@line
  \eql@hook@lineout
  &\omit
  \ifnum\eql@column@>\eql@totalcolumns@
    \global\eql@totalcolumns@\eql@column@
  \fi
```

TODO: not sure whether saving the last cell value makes sense, but rather not increase `\eql@totalcolumns@` because that will disable the fallback to lines mode. **TODO:** additional column in width table is accounted for in column table

```
  \ifdefined\eql@frame@cmd
    \advance\eql@column@\@ne
    \wd\eql@cellbox@\z@
    \eql@columns@measure@cell
  \fi
  \eql@columns@measure@tag
}
```

@columns@measure@tag

```
\def\eql@columns@measure@tag{%
  \eql@tagwidth@\z@
  \ifnum\eql@numbering@target@<\z@
    \if@eqnsw
      \eql@tagbox@make\eql@compose@measure
    \fi
  \fi
  \eql@widthdata@savetag
}
```

\eql@columns@measure

```
\def\eql@columns@measure{%
⟨dev⟩\eql@dev@enter\eql@columns@measure
  \eql@totalcolumns@\z@
  \eql@measure@init\eql@columns@measure@line@begin\eql@columns@measure@line@end

  \setbox\z@\vbox{\halign{%
    &%
      \global\advance\eql@column@\@ne
      \hfil
      \global\setbox\eql@cellbox@\hbox{%
        \@lign
        $\m@th\displaystyle
          \eql@hook@colin
          ##%
          \eql@class@innerleft
          \eql@hook@innerleft
        $%
      }%
      \global\eql@prevwidth@\wd\eql@cellbox@
      \eql@columns@measure@cell
    &%
      \global\advance\eql@column@\@ne
      \setbox\eql@cellbox@\hbox{%
        \@lign
        $\m@th\displaystyle
          \eql@hook@innerright
          \eql@class@innerright@sel
          ##%
          \eql@punct@apply@col
          \eql@hook@colout
        $%
      }%
      \eql@columns@measure@cell
      \hfil
    \crcr

    \noalign{%
      \eql@hook@blockbefore
    }%
    \eql@hook@blockin
    \eql@scan@body

    \ifvmode\else
      \eql@punct@apply@block
      \eql@hook@blockout
      \eql@display@endline
      \cr
    \fi
    \noalign{%
      \eql@hook@blockafter
    }%
```

TODO: note we also include the tag column as a backup

```
  \omit
  \eql@column@\@ne
  \eql@columns@completerow
  \cr
}}%

\eql@measure@close

\setbox\z@\vbox{%
  \unvbox\z@
  \unpenalty
  \global\setbox\@ne\lastbox
}%
\eql@totalwidth@\wd\@ne
```

TODO: why not recycle box contents altogether?! **TODO:** maybe add one column of safety?

```
\let\eql@colwidth@tab\@empty
\loop
  \setbox\@ne\hbox{%
```

```
      \unhbox\@ne
      \unskip
      \global\setbox\thr@@\lastbox
    }%
  \ifhbox\thr@@
    \eql@colwidth@save{\wd\thr@@}%
  \repeat
```

```
⟨dev⟩\eql@dev@leave\eql@columns@measure
}
```

N.3 Columns Placement

TODO: describe Make sure we have complete pairs of right and left adjusted columns, otherwise add a final empty column:

```
\def\eql@columns@adjust{%
  \ifodd\eql@totalcolumns@
    \advance\eql@totalcolumns@\@ne
  \fi
  \eql@adjust@calc@columns
}
```

N.4 Print

TODO: describe

mns@print@line@begin

```
\def\eql@columns@print@line@begin{%
⟨dev⟩\eql@dev{starting line \the\eql@row@}%
  \global\eql@column@\z@
  \global\eql@line@pos@\eql@marginleft@
  \global\eql@line@width@\z@
  \global\eql@line@avail@\eql@totalwidth@
  \global\eql@line@height@\z@
  \global\eql@line@depth@\z@
  \eql@numbering@print@line@begin
  \eql@hook@linein
}
```

l@columns@print@cell

```
\def\eql@columns@print@cell{%
  \eql@cellwidth@\wd\eql@cellbox@
  \ifodd\eql@column@
    \ifdefined\eql@frame@cmd
      \eql@frame@measure
      \advance\eql@cellwidth@\eql@frame@margin@
    \fi
    \dimen@\z@
  \else
    \advance\eql@cellwidth@-\eql@prevwidth@
```

draw a frame

```
    \ifdefined\eql@frame@cmd
      \eql@frame@measure
      \advance\eql@cellwidth@\eql@frame@margin@
      \advance\eql@prevwidth@\eql@frame@margin@
      \eql@frame@print
    \fi
```

update height and depth

```
    \ifdim\ht\eql@cellbox@>\eql@line@height@
      \global\eql@line@height@\ht\eql@cellbox@
    \fi
    \ifdim\dp\eql@cellbox@>\eql@line@depth@
      \global\eql@line@depth@\dp\eql@cellbox@
    \fi
```

print box

```
    \kern-\eql@prevwidth@
    \unhbox\eql@cellbox@
    \dimen@-\eql@cellwidth@
  \fi
```

enforce given width: hopefully measure was correct, but need a precise width for tag placement

```
  \advance\dimen@\eql@colwidth@get\eql@column@\relax
  \kern\dimen@
```

update available and used space

```
  \dimen@\eql@colwidth@get\eql@column@\relax
  \ifdim\eql@cellwidth@>\z@
    \ifdim\eql@line@width@=\z@
      \eql@line@avail@\eql@line@pos@
      \ifodd\eql@column@
        \advance\eql@line@avail@\dimen@
        \advance\eql@line@avail@-\eql@cellwidth@
      \fi
      \global\eql@line@avail@\eql@line@avail@
    \fi
    \eql@line@width@\eql@line@pos@
    \ifodd\eql@column@
      \advance\eql@line@width@\dimen@
    \else
      \advance\eql@line@width@\eql@cellwidth@
    \fi
    \global\eql@line@width@\eql@line@width@
  \fi
  \advance\eql@line@pos@\dimen@
  \ifodd\eql@column@\else
    \advance\eql@line@pos@\eql@colsep@
  \fi
  \global\eql@line@pos@\eql@line@pos@
}
```

```
\def\eql@columns@print@trailright{%
   &\omit
    \global\advance\eql@column@\@ne
    \eql@columns@print@cell
}
```

lumns@print@line@end

```
\def\eql@columns@print@line@end{%
  \eql@punct@apply@line
  \eql@hook@lineout
% \TODO add an even column with empty stuff if box processing deferred
  \ifodd\eql@column@
    \expandafter\eql@columns@print@trailright
  \fi
  \eql@columns@completerow
  \eql@columns@print@tag
}
```

ql@columns@print@tag

```
\def\eql@columns@print@tag{%
  \dimen@\eql@totalwidth@
  \advance\dimen@\eql@colsep@
  \kern-\dimen@
```

determine first line available space

```
  \eql@display@firstavail@set\eql@line@avail@
  \eql@columns@overfull
  \eql@numbering@print@line@eval
  \if@eqnsw
    \eql@tagbox@make\eql@compose@print
    \eql@tagging@tagaddbox
    \eql@tagbox@print@cell
  \else
    \eql@tagging@tagaddbox
    \kern\displaywidth
  \fi
}
```

\eql@columns@print

```
\def\eql@columns@print{%
⟨dev⟩\eql@dev@enter\eql@columns@print
  \eql@shape@align@disable
  \eql@display@halign@init\eql@columns@print@line@begin
  \eql@display@halign@letcr\eql@columns@print@line@end
  \tabskip\eql@marginleft@

  \halign{%
    &%
      \global\advance\eql@column@\@ne
      \hfil
      \global\setbox\eql@cellbox@\hbox{%
        \eql@strut@cell
        \@lign
        $\m@th\displaystyle
          \eql@hook@colin
          ##%
          \eql@class@innerleft
          \eql@hook@innerleft
          \eql@tagging@mathsave
        $%
        \eql@tagging@mathaddlast
      }%
      \global\eql@prevwidth@\wd\eql@cellbox@
      \eql@columns@print@cell
      \tabskip\z@skip
    &%
      \global\advance\eql@column@\@ne
      \setbox\eql@cellbox@\hbox{%
        \unhbox\eql@cellbox@
        \eql@strut@cell
        \@lign
        $\m@th\displaystyle
          \eql@hook@innerright
          \eql@class@innerright@sel
          ##%
          \eql@punct@apply@col
          \eql@hook@colout
          \eql@tagging@mathsave
        $%
        \eql@tagging@mathaddlast
      }%
      \eql@columns@print@cell
      \hfil
      \tabskip\eql@colsep@\relax
    \crcr

    \noalign{%
      \eql@display@halign@start
      \eql@numbering@print@block@begin
      \eql@hook@blockbefore
    }%
    \eql@hook@blockin
    \eql@scan@body
    \ifvmode\else
      \relax
      \eql@punct@apply@block
      \eql@hook@blockout
      \eql@display@endline
      \cr
    \fi
    \noalign{%
      \eql@hook@blockafter
      \eql@display@halign@end
⟨dev⟩\eql@dev@leave\eql@columns@print
    }%
    \eql@tagging@tablesavealign
  }%
}
```

O Interface

O.1 Scanning the Equation Body

The multi-line equatiuon environment must scan its 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. The mechanism and its description follows amsmath closely.

Token Register.

`\eql@scan@reg@` We start by defining a token register to hold the equation body.

```
\newtoks\eql@scan@reg@
```

`\eql@scan@body@dump` `eql@scan@body@rescan` `\eql@scan@body` The macro `\eql@scan@body@dump` dumps the equation body from the register so that we do not have to pass it around in arguments. The macro `\eql@scan@body@rescan` rescans the tokens so that special commands such as `\verb` can be processed properly. The register `\eql@scan@body` holds the currently selected mode of operation:

```
\def\eql@scan@body@dump{\the\eql@scan@reg@}
\def\eql@scan@body@rescan{%
  \expandafter\scantokens\expandafter{\the\eql@scan@reg@}}
\let\eql@scan@body\eql@scan@body@dump
```

`\eql@scan@addto` We define a macro to append to the token register `\eql@scan@reg@`:

```
\long\def\eql@scan@addto#1{\eql@scan@reg@\expandafter{\the\eql@scan@reg@#1}}
```

Environment Body. The following mechanism scans the contents of an environment taking into account nested environments that may be contained in the body.

`\eql@scan@env` The macro `\eql@scan@env` starts the scan for the `\end{...}` command of the current environment. The argument is a call-back macro to process the body in `\eql@scan@reg@`:

```
\def\eql@scan@env#1{%
⟨dev⟩\eql@dev@enter\eql@scan@env
  \def\eql@scan@end{#1\expandafter\end\expandafter{\@currenvir}}%
  \eql@scan@reg@{}\def\eql@scan@stack{b}%
```

We call `\eql@scan@env@iterate` which will scan until the next occurrence of `\end` and then count the number of occurrences of `\begin` before `\end` in `\eql@scan@stack`. If we simply called `\eql@scan@env@iterate` directly, the error message for an unwanted `\par` token (usually from a blank line) would refer to `\eql@scan@env@iterate` which would not be illuminating. We use a little finesse to get a more intelligible error message: We use the actual environment name as the name of the temporary function that is `\let` to `\eql@scan@env@iterate`:

```
  \edef\eql@scan@iterate{\expandafter\noexpand\csname\@currenvir\endcsname}%
  \expandafter\let\expandafter\eql@scan@env@org\eql@scan@iterate
  \ifdefined\eql@scan@par
    \expandafter\let\eql@scan@iterate\eql@scan@env@iterate
  \else
    \expandafter\let\eql@scan@iterate\eql@scan@env@iterate@nopar
  \fi
  \eql@scan@iterate
}
```

`eql@scan@env@iterate` `\eql@scan@env@iterate` 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 via `\eql@scan@env@count`. An 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:

```
\long\def\eql@scan@env@iterate#1\end#2{%
  \edef\eql@scan@stack{%
    \eql@scan@env@count#1\begin\end\expandafter\@gobble\eql@scan@stack}%
  \ifx\@empty\eql@scan@stack
```

```
    \@checkend{#2}%
    \eql@scan@addto{#1}%
    \expandafter\let\eql@scan@iterate\eql@scan@env@org
⟨dev⟩\eql@dev@leave\eql@scan@env
    \expandafter\eql@scan@end
  \else
    \eql@scan@addto{#1\end{#2}}%
    \expandafter\eql@scan@iterate
  \fi
}
```

an@env@iterate@nopar Version of `\eql@scan@env@iterate` which does not accept `\par` within the argument:

```
\def\eql@scan@env@iterate@nopar#1\end#2{\eql@scan@env@iterate#1\end{#2}}
```

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

```
\long\def\eql@scan@env@count#1\begin#2{%
  \ifx\end#2\else b\expandafter\eql@scan@env@count\fi
}
```

The call-back macro `\eql@scan@env@cancel` ignores the body as well as the end clause for the environment:

```
\def\eql@scan@env@cancel{%
  \@namedef{end\@currenvir}{\ignorespacesafterend}%
}
```

Square Brackets. The following is a version of the above mechanism that scans for an equation body enclosed by `\[...\]` paying attention to potential further instances of the square bracket enclosures contained in the body.

`\eql@scan@sqr` Start scanning for `\]`:

```
\def\eql@scan@sqr#1{%
⟨dev⟩\eql@dev@enter\eql@scan@sqr
  \def\eql@scan@end{#1\]}%
  \eql@scan@reg@{}\def\eql@scan@stack{b}%
  \let\eql@scan@sqr@org\[%\]
  \ifdefined\eql@scan@par
    \let\[\eql@scan@sqr@iterate%\]
  \else
    \let\[\eql@scan@sqr@iterate@nopar%\]
  \fi
  \[%\]
}
```

eql@scan@sqr@iterate Iterate until we find a balanced pairing of square brackets. Then call the call-back macro:

```
\long\def\eql@scan@sqr@iterate#1\]{%
  \edef\eql@scan@stack{%
    \eql@scan@sqr@count#1\[\]\expandafter\@gobble\eql@scan@stack}%
  \ifx\@empty\eql@scan@stack
    \let\[\eql@scan@sqr@org%\]
    \eql@scan@addto{#1}%
⟨dev⟩\eql@dev@leave\eql@scan@sqr
    \expandafter\eql@scan@end
  \else
    \eql@scan@addto{#1\]}%
    \expandafter\[%\]
  \fi
}
```

an@sqr@iterate@nopar Version of `\eql@scan@sqr@iterate` which does not accept `\par` within the argument:

```
\def\eql@scan@sqr@iterate@nopar#1\]{\eql@scan@sqr@iterate#1\]}
```

\eql@scan@sqr@count Push a ‘b’ for every encountered instance of ‘\[’:

```
\long\def\eql@scan@sqr@count#1\[#2{%\]
  \ifx\]#2\else b\expandafter\eql@scan@sqr@count\fi
}
```

l@scan@sqrang@cancel The call-back macro `\eql@scan@sqrang@cancel` ignores the body and the closing bracket:

```
\def\eql@scan@sqrang@cancel{\expandafter\ignorespaces\@gobble}
```

Angle Brackets. The following is another version of the mechanism which scans for an equation body enclosed by `\<...\>`.

\eql@scan@ang Start scanning for `\>`:

```
\def\eql@scan@ang#1{%
⟨dev⟩\eql@dev@enter\eql@scan@ang
  \def\eql@scan@end{#1\>}%
  \eql@scan@reg@{}\def\eql@scan@stack{b}%
  \let\eql@scan@ang@org\<%\>
  \ifdefined\eql@scan@par
    \let\<\eql@scan@ang@iterate%\>
  \else
    \let\<\eql@scan@ang@iterate@nopar%\>
  \fi
  \<%\>
}
```

eql@scan@ang@iterate Iterate until we find a balanced pairing of angle brackets:

```
\long\def\eql@scan@ang@iterate#1\>{%
  \edef\eql@scan@stack{%
    \eql@scan@ang@count#1\<\>\expandafter\@gobble\eql@scan@stack}%
  \ifx\@empty\eql@scan@stack
    \let\<\eql@scan@ang@org%\>
    \eql@scan@addto{#1}%
⟨dev⟩\eql@dev@leave\eql@scan@ang
    \expandafter\eql@scan@end
  \else
    \eql@scan@addto{#1\>}%
    \expandafter\<%\>
  \fi
}
```

an@ang@iterate@nopar Version of `\eql@scan@ang@iterate` which does not accept `\par` within the argument:

```
\def\eql@scan@ang@iterate@nopar#1\>{\eql@scan@ang@iterate#1\>}
```

\eql@scan@ang@count Push a ‘b’ for every encountered instance of ‘\<’:

```
\long\def\eql@scan@ang@count#1\<#2{%\>
  \ifx\>#2\else b\expandafter\eql@scan@ang@count\fi
}
```

O.2 Options Processing

ql@equations@testall The macro sequence started by \eql@equations@testall scans for optional arguments to the equation environments and appends them to the argument list using \eqnaddopt. The argument scheme is roughly `{ !t~ !t* !t! !o !e{@} }`. All arguments are scanned such that any spaces stop the scanning and such that any alignment markers ‘&’ cannot interfere:

```
\def\eql@equations@testall{\eql@equations@testtilde}
\def\eql@equations@testtilde#1{%
  \eql@ifnextgobble@tight~%
    {\eqnaddopt{lines}\eql@equations@testopt{#1}}%
    {\eql@equations@testopt{#1}}}
\def\eql@equations@testopt#1{%
  \eql@ifnextchar@tight[%]
    {\eql@equations@addopt{\eql@equations@testexcl{#1}}}%
    {\eql@equations@testexcl{#1}}}
\def\eql@equations@addopt#1[#2]{\eqnaddopt{#2}#1}
\def\eql@equations@testexcl#1{%
  \eql@ifnextgobble@tight!%
    {\eqnaddopt{donumber}\eql@equations@testat{#1}}%
    {\eql@equations@teststar{#1}}}
\def\eql@equations@teststar#1{%
  \eql@ifstar@tight%
    {\eqnaddopt{nonumber}\eql@equations@testat{#1}}%
    {\eql@equations@testat{#1}}}
\def\eql@equations@testat#1{%
  \eql@ifat@tight
    {\eql@equations@addlabel{#1}}%
    {#1}}
\def\eql@equations@addlabel#1#2{\eqnaddopt{label={#2}}#1}
```

equations@processopt The macro \eql@equations@processopt processes the options recevied by \eqnaddopt. First, clear several non-persistent registers (labels, tags, direct vertical spacing). Then process the arguments. Finally evaluate \eql@indent@val and \eql@tagsepmin@val and prevent main punctuation from being passed to nested environments:

```
\def\eql@equations@processopt{%
  \let\eql@blocklabel\@undefined
  \let\eql@blocklabelname\@undefined
  \let\eql@blocktag\@undefined
  \let\eql@skip@force@above\@undefined
  \let\eql@skip@force@below\@undefined
  \let\eql@skip@force@leave\@undefined
  \let\eql@display@linewidth\@undefined
  \let\eql@display@marginleft\@undefined
  \let\eql@display@marginright\@undefined
  \eql@abovespace@\z@skip
  \eql@belowspace@\z@skip
  \eql@displaybreak@prepen@\@MM
  \eql@displaybreak@postpen@\@MM
  \eql@frame@reset
  \eql@nextopt@process{equations}%
  \let\eql@punct@block\eql@punct@main
  \let\eql@punct@main\relax
  \eql@indent@\glueexpr\eql@indent@val\relax
  \eql@tagsepmin@\glueexpr\eql@tagsepmin@val\relax
}
```

O.3 Single-Line Main

In the following, we define the main routine for the single-line equation mode.

`\eql@single@cr` Cannot use line breaks, produce an error message:

```
\def\eql@single@cr{%
  \eql@error{Cannot use ‘\string\\’ within display equation.
    Please switch to equations environment}%
}
```

`\eql@single@start` Opening code for single-line equation. Capture current vertical mode, trigger pdf tagging, enter display math mode, initialise numbering scheme, backup current state of global registers, set native vs. manual equation tag mode, install error message for using \\. Hand over to mode-specific opening:

```
\def\eql@single@start{%
  \eql@display@enter
  \eql@tagging@start
  \eql@dollardollar@begin
  \eql@display@adjust
  \eql@numbering@mode@eval
  \eql@stack@save@single
  \eql@numbering@single@init
  \ifdefined\eql@single@crerror\else
    \let\\\eql@single@cr
  \fi
  \ifdefined\eql@single@native
    \let\eql@single@start@sel\eql@single@start@native
    \let\eql@single@end@sel\eql@single@end@native
  \else
    \let\eql@single@start@sel\eql@single@start@print
    \let\eql@single@end@sel\eql@single@end@print
  \fi
  \eql@single@start@sel
}
```

`\eql@single@end` Closing code for single-line equation. Apply punctuation for the block, perform mode-specific ending, restore global variables, end display math, indicate end to pdf tagging, return to vertical mode if desired:

```
\def\eql@single@end{%
  \eql@punct@apply@block
  \eql@hook@eqout
  \eql@single@end@sel
  \eql@stack@restore
  \eql@dollardollar@end
  \eql@tagging@end
  \eql@display@leave
}
```

`\eql@single@main` Combined opening, body and closing for pre-scanned body: **TODO:** is `\expandafter` needed? relic?

```
\def\eql@single@main{%
  \expandafter\eql@single@start
  \eql@scan@body
  \eql@single@end
}
```

`\eql@mode@single` Configure equations macros to single-line mode:

```
\def\eql@mode@single{%
  \ifdefined\eql@single@doscan
    \let\eql@equations@main\eql@single@main
    \let\eql@equations@end\@empty
  \else
    \let\eql@equations@main\@undefined
    \let\eql@equations@end\eql@single@end
  \fi
}
```

O.4 Multi-Line Main

`lti@mode@lines` (*bool*) Switch register for lines vs. columns mode:

```
\let\eql@multi@mode@lines\eql@false
```

`\eql@multi@main` Main routine for multi-line modes. Capture current vertical mode, trigger pdf tagging, enter display math mode, initialise numbering scheme, backup current state of global registers, initialise macros for use within equations: **TODO:** shove depends on lines vs columns

```
\def\eql@multi@main{%
  \eql@display@enter
  \eql@tagging@start
  \eql@dollardollar@begin
  \eql@display@adjust
  \eql@numbering@mode@eval
  \eql@stack@save@multi
  \ifdefined\eql@numbering@subeq@use
    \eql@numbering@subeq@init
  \fi
  \eql@display@init
  \let\intertext\eql@intertext
  \let\endintertext\endeql@intertext
  \eql@shape@align@enable
```

Now measure the given multi-line equations body:

```
  \ifdefined\eql@multi@mode@lines
    \eql@lines@measure
  \else
    \ifdefined\eql@ampproof@active
      \eql@ampproof
    \fi
    \eql@columns@measure
  \fi
```

If only a single equation number is used for subequation numbering, revert to normal equation numbering. If only a single column is used in columns mode, may fallback to lines mode. Switching from columns to lines mode, the width can be incorrect, expect only minor discrpancies, but for accurateness, should call `\eql@lines@measure`:

```
  \ifx\eql@numbering@subeq@use\@ne
    \eql@numbering@subeq@revert
  \fi
  \ifdefined\eql@multi@mode@lines\else
    \ifdefined\eql@multi@linesfallback
      \ifnum\eql@totalcolumns@=\@ne
        \let\eql@multi@mode@lines\eql@true
        \ifx\eql@multi@linesfallback\z@\else
          \eql@lines@measure
        \fi
      \fi
    \fi
  \fi
```

Adjust the multi-line equations body:

```
  \ifdefined\eql@multi@mode@lines
    \eql@lines@adjust
  \else
    \eql@columns@adjust
  \fi
```

Now print the multi-line equations body:

```
  \eql@display@print
  \eql@numbering@print@init
  \ifdefined\eql@multi@mode@lines
    \eql@lines@print
  \else
    \eql@columns@print
  \fi
  \eql@display@close
```

Close numbering, restore global variables, end display math, indicate end to pdf tagging, return to vertical mode if desired:

```
  \ifdefined\eql@numbering@subeq@use
    \eql@numbering@subeq@close
  \fi
  \eql@stack@restore
  \eql@dollardollar@end
  \eql@tagging@end
  \eql@display@leave
}
```

`\eql@mode@columns`
`\eql@mode@lines`

Configure equations macros to one of the two multi-line modes:

```
\def\eql@mode@columns{%
  \let\eql@equations@main\eql@multi@main
  \let\eql@equations@end\@empty
  \let\eql@multi@mode@lines\eql@false
}
\def\eql@mode@lines{%
  \let\eql@equations@main\eql@multi@main
  \let\eql@equations@end\@empty
  \let\eql@multi@mode@lines\eql@true
}
```

O.5 Equations Environment

We now declare the main environment and its symbolic versions.

Environment.

`equations` (*env.*) Declare the main equations environment. If already in math mode, fail and cancel the environment body. Otherwise scan for optional arguments and pass on to `\eql@equations@start`:

```
\newenvironment{equations}{%
⟨dev⟩\eql@dev@enterenv
  \ifmmode
    \eql@error@mathmode{\string\begin{\@currenvir}}%
    \expandafter\eql@scan@env\expandafter\eql@scan@env@cancel
  \else
    \expandafter\eql@ampprotect\expandafter\eql@equations@testall
      \expandafter\eql@equations@start
  \fi
}{%
  \eql@equations@end
  \ignorespacesafterend
⟨dev⟩\eql@dev@leaveenv
}
```

`\eql@equations@start` The macro `\eql@equations@start` first processes the arguments. Depending on the chosen mode of operation, scan the environment body passing on to `\eql@equations@main` or process a single-line equation via `\eql@single@start`:

```
\def\eql@equations@start{%
  \eql@equations@processopt
  \ifdefined\eql@equations@main
    \expandafter\eql@scan@env\expandafter\eql@equations@main
  \else
    \expandafter\eql@single@start
  \fi
}
```

Square Brackets.

`equations@sqr` (*env.*) Define a pseudo-environment `equations@sqr` such that `\@currenvir` may point to it when needed:

```
\newenvironment{equations@sqr}{}{}
```

`l@equations@sqr@open` Definition for '`\[`'. If already in math mode, ignore the enclosed contents. Otherwise add the default arguments `\eql@equations@sqr@opt`, enter the pseudo-environment, scan for optional arguments, and pass on to `\eql@equations@sqr@start`:

```
\protected\def\eql@equations@sqr@open{%
  \ifmmode
    \eql@error@mathmode{\string\[...\string\]}%
    \expandafter\eql@scan@sqr\expandafter\eql@scan@sqrang@cancel
  \else
⟨dev⟩\eql@dev@enter{\[...\string\]}%
    \expandafter\eqnaddopt\expandafter{\eql@equations@sqr@opt}%
    \begin{equations@sqr}%
    \let\]\eql@equations@sqr@close
    \expandafter\eql@ampprotect\expandafter\eql@equations@testall
      \expandafter\eql@equations@sqr@start
  \fi
}
```

@equations@sqr@start Process arguments. Depending on mode of operation, scan and process enclosed contents via `\eql@equations@main` or pass on to `\eql@single@start`:

```
\def\eql@equations@sqr@start{%
  \eql@equations@processopt
  \ifdefined\eql@equations@main
    \expandafter\eql@scan@sqr\expandafter\eql@equations@main
  \else
    \expandafter\eql@single@start
  \fi
}
```

@equations@sqr@close Definition for ‘`\]`’:

```
\protected\def\eql@equations@sqr@close{%
  \eql@equations@end
⟨dev⟩\eql@dev@leave{\[...\string\]}%
  \end{equations@sqr}%
  \ignorespaces
}
```

TODO: describe

\eql@sqr@open
\eql@sqr@close

```
\let\eql@sqr@open\eql@equations@sqr@open
\protected\def\eql@sqr@close{%
  \eql@error{'\string\]' may only close '\string\['}%\]
}
```

Angle Brackets.

`equations@ang` (*env.*) Define a pseudo-environment `equations@ang`:

```
\newenvironment{equations@ang}{}{}
\newenvironment{equationsbox@ang}{}{}
```

\eql@ang@open Definition for ‘`\<`’. Forward to `equationsbox` if in math mode, otherwise to `equations`:

```
\protected\def\eql@ang@open{%
⟨dev⟩\eql@dev@enter{\<...\string\>}%
  \ifmmode
    \expandafter\eqnaddopt\expandafter{\eql@box@ang@opt}%
    \begin{equationsbox@ang}%
    \let\>\eql@box@ang@close
    \expandafter\eql@ampprotect\expandafter\eql@box@testall
      \expandafter\eql@box@start
  \else
    \expandafter\eqnaddopt\expandafter{\eql@equations@ang@opt}%
    \begin{equations@ang}%
    \let\>\eql@equations@ang@close
    \expandafter\eql@ampprotect\expandafter\eql@equations@testall
      \expandafter\eql@equations@ang@start
  \fi
}
```

`\eql@ang@close` Definition for '`\>`': **TODO:** NOTE: `\protected` acts as `\relax` and starts a row in `\halign`, so we overwrite `\>` when starting.

```
\protected\def\eql@ang@close{%
  \eql@error{'\string\>' may only close '\string\<'}%
}
```

`@equations@ang@start` Process arguments and start handling the equation:

```
\def\eql@equations@ang@start{%
  \eql@equations@processopt
  \ifdefined\eql@equations@main
    \expandafter\eql@scan@ang\expandafter\eql@equations@main
  \else
    \expandafter\eql@single@start
  \fi
}
```

`@equations@ang@close` **TODO:** describe

```
\def\eql@equations@ang@close{%
  \eql@equations@end
  \end{equations@ang}%
⟨dev⟩\eql@dev@leave{\<...\string\>}%
  \ignorespaces
}
```

`\eql@box@ang@close` **TODO:** describe

```
\def\eql@box@ang@close{%
  \eql@box@end
  \end{equationsbox@ang}%
⟨dev⟩\eql@dev@leave{\<...\string\>}%
  \ignorespaces
}
```

P Options

The package uses the keyval mechanism to parse key-value pairs to specify adjustments to the behaviour of the equations environments:

```
\RequirePackage{keyval}
```

P.1 Selection Tools

`\eql@decide@select` Some parameter values take values in a given set, e.g. `true` vs. `false` or `left` vs. `right`. The macro `\eql@decide@select` is a general purpose selector. Arguments `#1` and `#2` describe the category and key which are used only towards error messages. Argument `#3` contains the value and argument `#4` is a list of values and corresponding actions in the format

`{{{`*val1a*`,`*val1b*`,`...`}{`*act1*`},{{`*val2a*`,`*val2b*`,`...`}{`*act2*`},`...`}`.

The (single) value `\relax` matches everything (can be used for handling generic values after specific ones). If no corresponding value is found in the list, an error message is invoked. Single expansion is applied to the list of values:

```
\def\eql@decide@relax{\@tempb:=\relax}
\def\eql@decide@select#1#2#3#4{%
  \def\@tempa{#3}%
  \let\@tempd\@undefined
  \@for\@tempc:=#4\do{%
    \ifdefined\@tempd\else
      \edef\@tempb{\noexpand\@tempb:=\expandafter\@firstoftwo\@tempc}%
      \ifx\@tempb\eql@decide@relax
        \def\@tempa{\relax}%
      \fi
      \expandafter\@for\@tempb\do{%
        \ifx\@tempa\@tempb
          \expandafter\expandafter\expandafter\def
          \expandafter\expandafter\expandafter\@tempd
          \expandafter\expandafter\expandafter{%
          \expandafter\@secondoftwo\@tempc}%
        \fi
      }%
    \fi
  }%
  \ifdefined\@tempd
    \@tempd
  \else
    \eql@error{undefined value ‘#3’ for option ‘#2’ of ‘#1’}%
  \fi
}
```

`\eql@decide@if` Decide between `true` and `false` or related pairs of values:

```
\def\eql@decide@true{on,true,yes,enabled,1}
\def\eql@decide@false{off,false,no,disabled,0}
\def\eql@decide@if#1#2#3#4#5{%
  \eql@decide@select{#1}{#2}{#3}{%
    {\eql@decide@true{#4}},%
    {\eql@decide@false{#5}}}}
```

`\eql@decide@bool` Store a boolean value into a conditional register:

```
\def\eql@decide@bool#1#2#3#4{%
  \eql@decide@if{#1}{#2}{#3}{\let#4\eql@true}{\let#4\eql@false}}
```

`ql@decide@abovebelow` Select between values ‘`above`’ or ‘`below`’ or both: execute the corresponding code provided in the latter two arguments:

```
\def\eql@decide@abovebelow#1#2#3#4#5{%
  \eql@decide@select{#1}{#2}{#3}{%
    {,abovebelow,both,tb}{#4#5},%
    {above,top,t}{#4},%
    {below,bottom,b}{#5}}}
```

`eql@decide@situation` Select a particular vertical spacing situation and store it in the macro `#4`:

```
\def\eql@decide@situation#1#2#3#4{%
  \eql@decide@select{#1}{#2}{#3}{%
    {{long}{\def#4{0}}},%
    {{short}{\def#4{1}}},%
    {{cont}{\def#4{2}}},%
    {{par}{\def#4{3}}},%
    {{top}{\def#4{4}}},%
    {{noskip}{\def#4{5}}},%
    {{medskip}{\def#4{6}}}}}
```

P.2 Declaration Code

`\eql@define@key` For convenience, we define a wrapper for keyval's `\define@key` which accepts lists of categories and keys. We prepend the prefix `eql@` to all our categories so that we can hide it from the user in error messages:

```
\def\eql@define@key#1#2{%
  \eql@ifnextchar@loose[%]
    {\eql@definekey@opt{#1}{#2}}%
    {\eql@definekey@noopt{#1}{#2}}%
}
\def\eql@definekey@noopt#1#2#3{\eql@definekey@for{#1}{#2}{{#3}}}
\def\eql@definekey@opt#1#2[#3]#4{\eql@definekey@for{#1}{#2}{[#3]{#4}}}
\def\eql@definekey@for#1#2#3{%
  \def\eql@for@fn##1##2##3{\define@key{eql@##3}{##2}#3}%
  \edef\eql@for@vara{\noexpand\eql@for@vara:=#1}%
  \expandafter\@for\eql@for@vara\do{%
    \edef\eql@for@varb{\noexpand\eql@for@varb:=#2}%
    \expandafter\@for\eql@for@varb\do{%
      \edef\eql@for@call##1{%
        \noexpand\eql@for@fn{##1}{\eql@for@varb}{\eql@for@vara}}%
      \eql@for@call{##1}%
    }%
  }%
}
```

`\eql@setkeys` Our wrapper of keyval's `\setkeys` prepends the prefix `eql@` to the category, and it expands the list argument once:

```
\def\eql@setkeys#1#2{%
  \def\eql@tmp{\setkeys{eql@#1}}%
  \expandafter\eql@tmp\expandafter{#2}%
}
```

`\eql@nextopt` `\eql@nextopt@process` `\eqnaddopt` It can be convenient to add arguments to the following equations environment, e.g. towards defining modifier macros:

```
\let\eql@nextopt\@empty
\def\eql@nextopt@process#1{%
⟨dev⟩\eql@dev@start\eql@nextopt@process
  \eql@setkeys{#1}\eql@nextopt
  \let\eql@tagging@opt\eql@nextopt
  \global\let\eql@nextopt\@empty
}
\newcommand{\eqnaddopt}[1]{%
  \expandafter\def\expandafter\eql@nextopt\expandafter{\eql@nextopt,#1}}
```

P.3 Options Declarations

We now declare all key-value pairs for options sorted by their category.

Modes for Equations Box Environment. Declare horizontal and vertical alignment modes for the boxed equations environment. Also declare spacing of columns:

```
\eql@define@key{equationsbox}{gathered,gather,ga,lines,ln,\string~}[]{%
  \eql@mode@stacked}
\eql@define@key{equationsbox}{aligned,align,al,columns,col,@}[]{%
  \eql@mode@aligned}
\eql@define@key{equationsbox}{top,t}[]{\let\eql@box@box\vtop}
\eql@define@key{equationsbox}{center,c}[]{\let\eql@box@box\vcenter}
\eql@define@key{equationsbox}{bottom,b}[]{\let\eql@box@box\vbox}
\eql@define@key{equationsbox}{frame}[\fbox]{%
  \def\eql@box@frame{#1}%
  \ifx\eql@box@frame\@empty\let\eql@box@frame\@firstofone\fi}
\eql@define@key{equationsbox}{wrap}[]{\eql@box@wrap#1}
\eql@define@key{setup}{boxangopt}[]{%
  \def\eql@box@ang@opt{columns,#1}}
```

Modes for Equations Environment. Declare modes and switches for the equations environment:

```
\eql@define@key{equations}{equation,eq,single,1}[]{\eql@mode@single}
\eql@define@key{equations}{gathered,gather,ga,lines,ln,\string~}[]{%
  \eql@mode@lines}
\eql@define@key{equations}{aligned,align,al,columns,col,@}[]{%
  \eql@mode@columns}
\eql@define@key{equations}{native}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@single@native%
  \ifdefined\eql@single@native\let\eql@layoutleft\eql@false\fi}
\eql@define@key{setup}{native}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@single@native}
\eql@define@key{setup}{scanequation}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@single@doscan}
\eql@define@key{setup}{sqropt}[]{%
  \def\eql@equations@sqr@opt{equation,#1}}
\eql@define@key{setup}{angopt}[]{%
  \def\eql@equations@ang@opt{columns,#1}}
```

Vertical Spacing. Settings concerning the spacing of lines: **TODO:** set at end of env only!

```
\def\eql@keycat{equations,equationsbox,setup}
\eql@define@key\eql@keycat{spread}{\def\eql@spread@val{#1}}
\eql@define@key\eql@keycat{strut}[true]{\eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\let\eql@strut@cell\relax\let\eql@strut@tag\relax}},%
    {{cell}{\let\eql@strut@cell\eql@strut\let\eql@strut@tag\relax}},%
    {{tag}{\let\eql@strut@cell\relax\let\eql@strut@tag\eql@strut}},%
    {\eql@decide@true
      {\let\eql@strut@cell\eql@strut\let\eql@strut@tag\eql@strut}}}}
\eql@define@key{setup}{strutdepth}{\def\eql@strut@depth{#1}}
```

Settings concerning page breaks:

```
\eql@define@key{equations}{prebreak}[4]{\eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@true{\eql@displaybreak@pre4}},%
    {{force}{\eql@displaybreak@pre4}},%
    {{high}{\eql@displaybreak@pre3}},%
    {{med,medium}{\eql@displaybreak@pre2}},%
    {{low}{\eql@displaybreak@pre1}},%
    {\eql@decide@false{\eql@displaybreak@pre0}},%
    {{default,inherit,0}{\eql@displaybreak@pre\m@ne}},%
    {\relax{\eql@displaybreak@pre{#1}}}}}
\eql@define@key{equations}{postbreak}[4]{\eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@true{\eql@displaybreak@post4}},%
    {{force}{\eql@displaybreak@post4}},%
    {{high}{\eql@displaybreak@post3}},%
    {{med,medium}{\eql@displaybreak@post2}},%
    {{low}{\eql@displaybreak@post1}},%
    {\eql@decide@false{\eql@displaybreak@post0}},%
    {{default,inherit,}{\eql@displaybreak@post\m@ne}},%
    {\relax{\eql@displaybreak@post{#1}}}}}
\eql@define@key{equations,setup}{allowbreaks,allowdisplaybreaks}[4]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{full}{\eql@displaybreak@inter4}},%
    {{high}{\eql@displaybreak@inter3}},%
    {{med,medium}{\eql@displaybreak@inter2}},%
    {{low}{\eql@displaybreak@inter1}},%
    {\eql@decide@false{\eql@displaybreak@inter\z@}},%
    {\relax{\eql@displaybreak@inter{#1}}}}}
\eql@define@key{equations}{prepenalty}{%
  \eql@displaybreak@prepen@\numexpr#1\relax}
\eql@define@key{equations}{postpenalty}{%
  \eql@displaybreak@postpen@\numexpr#1\relax}
\eql@define@key{equations,setup}{interpenalty}{%
  \interdisplaylinepenalty\numexpr#1\relax}
```

Settings to specify the apparent height and depth of equations:

```
\eql@define@key\eql@keycat{displayheight}[strut]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\let\eql@display@height\@undefined}},%
    {{strut}{\def\eql@display@height{\ht\eql@strutbox@}}},%
    {\relax{\def\eql@display@height{#1}}}}}
\eql@define@key\eql@keycat{displaydepth}[strut]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\let\eql@display@depth\@undefined}},%
    {{strut}{\def\eql@display@depth{\dp\eql@strutbox@}}},%
    {\relax{\def\eql@display@depth{#1}}}}}
```

Override vertical spacing situation: **TODO:** short should just apply to above?! or as far as short would apply...

```
\eql@define@key{equations}{noskip}[]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@above{5}}%
    {\def\eql@skip@force@below{5}}}
\eql@define@key{equations}{short}[above]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@above{1}}%
    {\def\eql@skip@force@below{1}}}
\eql@define@key{equations}{long}[]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@above{0}}%
    {\def\eql@skip@force@below{0}}}
\eql@define@key{equations}{medskip}[]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@above{6}}%
    {\def\eql@skip@force@below{6}}}
\eql@define@key{equations}{par}[par]{%
```

```
  \eql@decide@select{#3}{#2}{#1}{%
    {{default,}{\let\eql@skip@force@leave\@undefined}},%
    {{cont,hmode}{\let\eql@skip@force@leave\z@}},%
    {{par,vmode}{\let\eql@skip@force@leave\@ne
      \ifdefined\eql@skip@force@below\else
        \def\eql@skip@force@below{3}%
      \fi}},%
    {{top}{\let\eql@skip@force@leave\tw@
      \ifdefined\eql@skip@force@below\else
        \def\eql@skip@force@below{4}
      \fi}}}}
```

Specify vertical spacing explicitly:

```
\eql@define@key{equations}{skip}{%
  \def\eql@skip@force@above{7}%
  \def\eql@skip@custom@above{#1}%
  \let\eql@skip@force@below\eql@skip@force@above
  \let\eql@skip@custom@below\eql@skip@custom@above}
\eql@define@key{equations}{aboveskip}{%
  \def\eql@skip@force@above{7}%
  \def\eql@skip@custom@above{#1}}
\eql@define@key{equations}{belowskip}{%
  \def\eql@skip@force@below{7}%
  \def\eql@skip@custom@below{#1}}
\eql@define@key{equations}{abovespace}{%
  \advance\eql@abovespace@\glueexpr#1\relax}
\eql@define@key{equations}{belowspace}{%
  \advance\eql@belowspace@\glueexpr#1\relax}
```

Vertical spacing for `intertext`:

```
\eql@define@key{intertext}{skip}{%
  \def\eql@skip@force@above{7}%
  \def\eql@skip@custom@above{#1}%
  \let\eql@skip@force@below\eql@skip@force@above
  \let\eql@skip@custom@below\eql@skip@custom@above}
\eql@define@key{intertext}{aboveskip}{%
  \def\eql@skip@force@below{7}%
  \def\eql@skip@custom@below{#1}}
\eql@define@key{intertext}{belowskip}{%
  \def\eql@skip@force@above{7}%
  \def\eql@skip@custom@above{#1}}
\eql@define@key{intertext}{noskip}[]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@below{5}}%
    {\def\eql@skip@force@above{5}}}
\eql@define@key{intertext}{short}[]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@below{1}}%
    {\def\eql@skip@force@above{1}}}
\eql@define@key{intertext}{long}[]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@below{0}}%
    {\def\eql@skip@force@above{0}}}
\eql@define@key{intertext}{medskip}[]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@below{6}}%
    {\def\eql@skip@force@above{6}}}
```

Configure general vertical spacing behaviour for various situations:

```
\eql@define@key{setup}{skip,longskip}{%
  \abovedisplayskip\glueexpr#1\relax
  \belowdisplayskip\abovedisplayskip
  \def\eql@skip@long@above{#1}%
  \let\eql@skip@long@below\eql@skip@long@above}
\eql@define@key{setup}{aboveskip,abovelongskip}{%
  \abovedisplayskip\glueexpr#1\relax
  \def\eql@skip@long@above{#1}}
\eql@define@key{setup}{belowskip,belowlongskip}{%
  \belowdisplayskip\glueexpr#1\relax
  \def\eql@skip@long@below{#1}}
\eql@define@key{setup}{aboveshortskip}{%
  \abovedisplayshortskip\glueexpr#1\relax
  \def\eql@skip@short@above{#1}}
\eql@define@key{setup}{belowshortskip}{%
  \belowdisplayshortskip\glueexpr#1\relax
  \def\eql@skip@short@below{#1}}
\eql@define@key{setup}{tagskip}{%
  \def\eql@skip@tag@above{#1}%
  \let\eql@skip@tag@below\eql@skip@tag@above}
\eql@define@key{setup}{abovetagskip}{%
  \def\eql@skip@tag@above{#1}}
\eql@define@key{setup}{belowtagskip}{%
  \def\eql@skip@tag@below{#1}}
\eql@define@key{setup}{medskip}{%
  \def\eql@skip@med@above{#1}%
  \let\eql@skip@med@below\eql@skip@med@above}
\eql@define@key{setup}{abovemedskip}{%
  \def\eql@skip@med@above{#1}}
\eql@define@key{setup}{belowmedskip}{%
  \def\eql@skip@med@below{#1}}
\eql@define@key{setup}{medtagskip}{%
  \def\eql@skip@medtag@above{#1}%
  \let\eql@skip@medtag@below\eql@skip@medtag@above}
\eql@define@key{setup}{abovemedtagskip}{%
  \def\eql@skip@medtag@above{#1}}
\eql@define@key{setup}{belowmedtagskip}{%
  \def\eql@skip@medtag@below{#1}}
\eql@define@key{setup}{abovetopskip}{%
  \def\eql@skip@top@above{#1}}
\eql@define@key{setup}{belowtopskip}{%
  \def\eql@skip@top@below{#1}}
\eql@define@key{setup}{aboveparskip}{%
  \def\eql@skip@par@above{#1}}
\eql@define@key{setup}{belowparskip}{%
  \def\eql@skip@par@below{#1}}
\eql@define@key{setup}{abovepartagskip}{%
  \def\eql@skip@partag@above{#1}}
\eql@define@key{setup}{belowpartagskip}{%
  \def\eql@skip@partag@below{#1}}
\eql@define@key{setup}{abovecontskip}{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{hide}{\def\eql@skip@cont@above{\eql@spread@val-\eql@skip@long@below}}},%
    {\relax{\def\eql@skip@cont@above{#1}}}}}
\eql@define@key{setup}{belowcontskip}{%
  \def\eql@skip@cont@below{#1}}
\eql@define@key{setup}{shortmode}{%
```

```
  \eql@decide@select{#3}{#2}{#1}{%
    {{off,never,no}{\def\eql@skip@mode@short{0}}},%
    {{above,neverbelow,notbelow,belowoff}{\def\eql@skip@mode@short{1}}},%
    {{belowone,belowsingle}{\def\eql@skip@mode@short{2}}},%
    {{belowall,always,on}{\def\eql@skip@mode@short{3}}}}}
\eql@define@key{setup}{abovecontmode}{%
  \eql@decide@situation{#3}{#2}{#1}\eql@skip@mode@cont@above}
\eql@define@key{setup}{belowcontmode}{%
  \eql@decide@situation{#3}{#2}{#1}\eql@skip@mode@cont@below}
\eql@define@key{setup}{aboveparmode}{%
  \eql@decide@situation{#3}{#2}{#1}\eql@skip@mode@par@above}
\eql@define@key{setup}{belowparmode}{%
  \eql@decide@situation{#3}{#2}{#1}\eql@skip@mode@par@below}
\eql@define@key{setup}{abovetopmode}{%
  \eql@decide@situation{#3}{#2}{#1}\eql@skip@mode@top@above}
\eql@define@key{setup}{belowtopmode}{%
  \eql@decide@situation{#3}{#2}{#1}\eql@skip@mode@top@below}
```

Labels and Tag Declaration. Specify label and tag for equations and subequations:

```
\def\eql@keycat{equations,subequations}
\eql@define@key\eql@keycat{label}{\eql@blocklabel@set{#1}}
\eql@define@key\eql@keycat{labelname}{\eql@blocklabelname@set{#1}}
\eql@define@key\eql@keycat{tag}{\eql@blocktag@set\@ne{#1}}
\eql@define@key\eql@keycat{tag*}{\eql@blocktag@set\z@{#1}}
\eql@define@key{setup}{labelname}{\protected@edef\eql@labelname@generic{#1}}
\eql@define@key{setup}{autolabel}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@numbering@autolabel}
\eql@define@key{setup}{autotag}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@numbering@autotag}
```

Tag Spacing. Configure horizontal spacing for equation tags:

```
\def\eql@keycat{equations,setup}
\eql@define@key\eql@keycat{tagmargin}[auto]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\let\eql@tagmargin@val\@undefined}},%
    {{auto}{\let\eql@tagmargin@val\@undefined}},%
    {\relax{\def\eql@tagmargin@val{#1}}}}}
\eql@define@key\eql@keycat{tagmargin*}{%
  \settowidth\dimen@{#1}\edef\eql@tagmargin@val{\the\dimen@}}
\eql@define@key\eql@keycat{tagmarginratio}{%
  \eql@tagmargin@ratio@\dimexpr#1pt\relax}
\eql@define@key\eql@keycat{tagmarginthreshold}{%
  \def\eql@tagmargin@threshold{#1}}
\eql@define@key\eql@keycat{mintagsep}{\def\eql@tagsepmin@val{#1}}
\eql@define@key\eql@keycat{mintagwidth}{%
  \settowidth\dimen@{#1}\edef\eql@tagsepmin@val{\the\dimen@}}
\eql@define@key\eql@keycat{mintagwidth*}{\settowidth\eql@tagwidthmin@{#1}}
```

Tag Layout. Configure methods to declare equation tag layout:

```
\eql@define@key{setup}{tagbox,taglayout}{\eql@tag@setbox{#1}}
\eql@define@key{setup}{tagbox*,taglayout*}{\eql@tag@setbox@{#1}}
\eql@define@key{setup}{tagform}{\eql@tag@setform#1}
\eql@define@key{setup}{tagform*}{\eql@tag@setform@{#1}}
\eql@define@key{setup}{subeqtemplate}{%
```

```
  \def\eql@subequations@template####1####2{#1}%
  \expandafter\def\expandafter\eql@subequations@template\expandafter{%
    \eql@subequations@template\theparentequation{equation}}%
}
```

Equation Numbering. Configure equation numbering schemes:

```
\def\eql@keycat{equations,setup}
\eql@define@key\eql@keycat{numberline,numline,n}[all]{%
  \eql@numbering@set{##1}}
\eql@define@key\eql@keycat{nonumber,nn,*}[]{%
  \let\eql@numbering@active\eql@false}
\eql@define@key\eql@keycat{donumber,dn,!}[]{%
  \let\eql@numbering@active\eql@true}
\eql@define@key\eql@keycat{number,num}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@numbering@active}
\eql@define@key\eql@keycat{tagsleft,leqno}[]{\let\eql@tagsleft\eql@true}
\eql@define@key\eql@keycat{tagsright,reqno}[]{\let\eql@tagsleft\eql@false}
\eql@define@key\eql@keycat{tags,eqno}{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{right,r}{\let\eql@tagsleft\eql@false}},%
    {{left,l}{\let\eql@tagsleft\eql@true}}}}
\eql@define@key\eql@keycat{bestlineauto}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@numbering@best@auto}
```

Horizontal Layout. Configure horizontal alignment mode and margin for left alignment:

```
\def\eql@keycat{equations,setup}
\eql@define@key\eql@keycat{layout}{\eql@decide@select{#3}{#2}{#1}{%
  {{center,c}{\let\eql@layoutleft\eql@false}},%
  {{left,l}{\let\eql@layoutleft\eql@true}}}}
\eql@define@key\eql@keycat{center}[]{\let\eql@layoutleft\eql@false}
\eql@define@key\eql@keycat{flushleft,left}[]{\let\eql@layoutleft\eql@true}
\eql@define@key\eql@keycat{leftmargin}{\def\eql@layoutleftmargin{#1}}
\eql@define@key\eql@keycat{leftmargin*}{%
  \settowidth\dimen@{#1}\edef\eql@layoutleftmargin{\the\dimen@}}
\eql@define@key\eql@keycat{minleftmargin}{%
  \def\eql@layoutleftmarginmin{#1}}
\eql@define@key\eql@keycat{maxleftmargin}{%
  \eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\def\eql@layoutleftmarginmax{.5\maxdimen}}},%
    {\relax{\def\eql@layoutleftmarginmax{#1}}}}}
```

```
\def\eql@keycat{equations,equationsbox}
\eql@define@key\eql@keycat{margin}{%
  \def\eql@display@marginleft{#1}\def\eql@display@marginright{#1}}
\eql@define@key\eql@keycat{marginleft}{\def\eql@display@marginleft{#1}}
\eql@define@key\eql@keycat{marginright}{\def\eql@display@marginright{#1}}
\eql@define@key{equations}{linewidth,width}{\def\eql@display@linewidth{#1}}
```

Horizontal Spacing and Columns. Configure column spacing and compression threshold:

```
\def\eql@keycat{equations,setup}
\eql@define@key\eql@keycat{alignshrink}{\eql@decide@select{#3}{#2}{#1}{%
    {{max,full}{\eql@alignbadness@\inf@bad}},%
```

```
  {{high}{\eql@alignbadness@54\relax}},%
  {{med,medium}{\eql@alignbadness@18\relax}},%
  {{low}{\eql@alignbadness@6\relax}},%
  {\eql@decide@false{\eql@alignbadness@\z@}}}}
\eql@define@key\eql@keycat{tagshrink}{\eql@decide@select{#3}{#2}{#1}{%
  {{max,full}{\eql@tagbadness@\inf@bad}},%
  {{high}{\eql@tagbadness@54\relax}},%
  {{med,medium}{\eql@tagbadness@18\relax}},%
  {{low}{\eql@tagbadness@6\relax}},%
  {{max,full}{\eql@tagbadness@\inf@bad}},%
  {\eql@decide@false{\eql@tagbadness@\z@}}}}
\eql@define@key\eql@keycat{alignbadness}{\eql@alignbadness@\numexpr#1\relax}
\eql@define@key\eql@keycat{tagbadness}{\eql@tagbadness@\numexpr#1\relax}
\eql@define@key\eql@keycat{mincolsep}{\eql@decide@select{#3}{#2}{#1}{%
  {\eql@decide@false{\def\eql@colsepmin@val{0pt}}},%
  {\relax{\def\eql@colsepmin@val{#1}}}}}
\eql@define@key\eql@keycat{maxcolsep}{\eql@decide@select{#3}{#2}{#1}{%
  {\eql@decide@false{\def\eql@colsepmax@val{.5\maxdimen}}},%
  {\relax{\def\eql@colsepmax@val{#1}}}}}
\eql@define@key\eql@keycat{fulllength}[true]{%
 \eql@decide@bool{#3}{#2}{#1}\eql@columns@fulllength}
```

```
\eql@define@key{equationsbox,setup}{colsep}{\def\eql@box@colsep{#1}}
```

Horizontal Shape. Configure horizontal alignment schemes:

```
\def\eql@keycat{equations,equationsbox,setup}
\eql@define@key\eql@keycat{shape}[default]{\eql@shape@set{#1}}
\eql@define@key\eql@keycat{padding,pad}[indent]{%
 \eql@decide@select{#3}{#2}{#1}{%
  {{max}{\let\eql@paddingleft@val\@undefined}},%
  {{indent}{\def\eql@paddingleft@val{\eql@indent@val}}},%
  {\eql@decide@false{\def\eql@paddingleft@val{0pt}}},%
  {\relax{\def\eql@paddingleft@val{#1}}}}%
 \let\eql@paddingright@val\eql@paddingleft@val}
\eql@define@key\eql@keycat{padleft}[indent]{%
 \eql@decide@select{#3}{#2}{#1}{%
  {{max}{\let\eql@paddingleft@val\@undefined}},%
  {{indent}{\def\eql@paddingleft@val{\eql@indent@val}}},%
  {\eql@decide@false{\def\eql@paddingleft@val{0pt}}},%
  {\relax{\def\eql@paddingleft@val{#1}}}}}
\eql@define@key\eql@keycat{padright}[indent]{%
 \eql@decide@select{#3}{#2}{#1}{%
  {{max}{\let\eql@paddingright@val\@undefined}},%
  {{indent}{\def\eql@paddingright@val{\eql@indent@val}}},%
  {\eql@decide@false{\def\eql@paddingright@val{0pt}}},%
  {\relax{\def\eql@paddingright@val{#1}}}}}
\eql@define@key\eql@keycat{indent}[2em]{%
 \def\eql@indent@val{#1}}
```

Math Classes at Alignment. Configure math classes at alignment marker:

```
\def\eql@keycat{equations,equationsbox,setup}
\eql@define@key\eql@keycat{classout}{\eql@class@innerleft@set{#1}}
\eql@define@key\eql@keycat{classin}{\eql@class@innerright@set{#1}}
\eql@define@key\eql@keycat{classlead,classin*}{\eql@class@innerlead@set{#1}}
\eql@define@key\eql@keycat{ampeq}[]{\eql@class@ampeq}
\eql@define@key\eql@keycat{eqamp}[]{\eql@class@eqamp}
```

```
\eql@define@key\eql@keycat{class}{\eql@decide@select{#3}{#2}{#1}{%
  {{ampeq,amprel,eqafter,beforerel}\eql@class@ampeq},%
  {{eqamp,relamp,eqbefore,afterrel}\eql@class@eqamp}}}
```

Punctuation. Configure punctuation defaults:

```
\let\eql@punct@main\relax
\def\eql@keycat{equations,equationsbox,setup}
\eql@define@key\eql@keycat{punctsep}[\,]{\def\eql@punct@sep{#1}}
\eql@define@key\eql@keycat{punct}[.]{\def\eql@punct@main{#1}}
\eql@define@key\eql@keycat{punctline}[,]{\def\eql@punct@line{#1}}
\eql@define@key\eql@keycat{punctcol}[,]{\def\eql@punct@col{#1}}
\eql@define@key\eql@keycat{punct*}[]{\let\eql@punct@main\relax}
\eql@define@key\eql@keycat{punctline*}[]{\let\eql@punct@line\relax}
\eql@define@key\eql@keycat{punctcol*}[]{\let\eql@punct@col\relax}
```

Alternative Content Description. Alternative content description for accessibility or documentation purposes: **TODO:** implement in pdf tagging

```
\eql@define@key{equations,equationsbox}{alt}{}
```

Global Switches. Set global switches:

```
\let\eql@multi@linesfallback\eql@false
\let\eql@scan@par\eql@false
\let\eql@single@crerror\eql@false
\let\eql@ampproof@active\eql@false
```

```
\eql@define@key{setup}{linesfallback}[true]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\let\eql@multi@linesfallback\eql@false}},%
    {\eql@decide@true{\let\eql@multi@linesfallback\eql@true}},%
    {{reuse,lean}{\let\eql@multi@linesfallback\z@}},%
    {{measure,full}{\let\eql@multi@linesfallback\eql@true}}}}
\eql@define@key{setup}{ampproof}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@ampproof@active}
\eql@define@key{setup}{crerror}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@single@crerror}
\eql@define@key{equations,setup}{rescan}[true]{%
  \eql@decide@if{#3}{#2}{#1}%
    {\let\eql@scan@body\eql@scan@body@rescan}%
    {\let\eql@scan@body\eql@scan@body@dump}}
\eql@define@key{equations,equationsbox,setup}{scanpar}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@scan@par}
\eql@define@key{setup}{defaults}{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{classic}{\eql@defaults@classic}},%
    {{eqnlines}{\eql@defaults@eqnlines}}}}
```

Package Options. Declare choices available at loading of package only:

```
\let\eql@provide@opt@equation\eql@true
\let\eql@provide@opt@amsmathends\eql@true
\let\eql@provide@opt@backup\eql@false
\let\eql@provide@opt@amsmath\eql@true
\let\eql@provide@opt@ang\eql@true
\let\eql@provide@opt@eqref\eql@true
```

```
\eql@define@key{setup}{equation}[true]{%
  \eql@error@packageoption{#2}%
  \eql@decide@bool{#3}{#2}{#1}\eql@provide@opt@equation}
\eql@define@key{setup}{amsmathends}[true]{%
  \eql@error@packageoption{#2}%
  \eql@decide@bool{#3}{#2}{#1}\eql@provide@opt@amsmathends}
\eql@define@key{setup}{backup}[true]{%
  \eql@error@packageoption{#2}%
  \eql@decide@bool{#3}{#2}{#1}\eql@provide@opt@backup}
\eql@define@key{setup}{amsmath}[true]{%
  \eql@error@packageoption{#2}%
  \eql@decide@bool{#3}{#2}{#1}\eql@provide@opt@amsmath}
\eql@define@key{setup}{ang}[true]{%
  \eql@error@packageoption{#2}%
  \eql@decide@bool{#3}{#2}{#1}\eql@provide@opt@ang}
\eql@define@key{setup}{eqref}[true]{%
  \eql@error@packageoption{#2}%
  \eql@decide@bool{#3}{#2}{#1}\eql@provide@opt@eqref}
```

P.4 Parameter Presets

The package offers two parameter presets which lead to somewhat different layout. Instead of setting the internal parameters directly, we expose them as public settings so that they are easier to read and such that individual settings can be used to compose own layouts.

eql@defaults@classic The preset `classic` aims to reproduce the TeX, LaTeX and amsmath layout closely. These presets mostly use fixed dimensions:

```
\def\eql@defaults@classic{%
  \eqnlinesset{numberline=all}%
  \eqnlinesset{mintagsep={.5\fontdimen6\textfont2\relax}}%
  \eqnlinesset{maxcolsep=off}%
  \eqnlinesset{spread={\jot}}%
  \eqnlinesset{tagmargin}%
  \eqnlinesset{tagmarginratio=1}%
  \eqnlinesset{tagmarginthreshold=0.5}%
  \eqnlinesset{leftmargin={\leftmargini}}%
  \eqnlinesset{padding=max}%
  \eqnlinesset{bestlineauto=off}%
  \eqnlinesset{displayheight=off}%
  \eqnlinesset{displaydepth=off}%
  \eqnlinesset{shortmode=belowsingle}%
  \eqnlinesset{abovecontmode=short}%
  \eqnlinesset{belowcontmode=short}%
  \eqnlinesset{aboveparmode=long}%
  \eqnlinesset{belowparmode=long}%
  \eqnlinesset{abovetopmode=long}%
  \eqnlinesset{belowtopmode=long}%
  \eqnlinesset{abovelongskip={\abovedisplayskip}}%
  \eqnlinesset{belowlongskip={\belowdisplayskip}}%
  \eqnlinesset{aboveshortskip={\abovedisplayshortskip}}%
  \eqnlinesset{belowshortskip={\belowdisplayshortskip}}%
  \eqnlinesset{abovemedskip={.5\abovedisplayskip}}%
  \eqnlinesset{belowmedskip={.5\belowdisplayskip}}%
  \eqnlinesset{abovecontskip=0pt}%
  \eqnlinesset{belowcontskip=0pt}%
  \eqnlinesset{aboveparskip=0pt}%
  \eqnlinesset{belowparskip=0pt}%
  \eqnlinesset{abovetopskip=0pt}%
  \eqnlinesset{belowtopskip=0pt}%
  \eqnlinesset{abovetagskip=0pt}%
  \eqnlinesset{belowtagskip=0pt}%
  \eqnlinesset{abovemedtagskip=0pt}%
  \eqnlinesset{belowmedtagskip=0pt}%
  \eqnlinesset{abovepartagskip=0pt}%
  \eqnlinesset{belowpartagskip=0pt}%
  \eqnlinesset{crerror=false}%
  \eqnlinesset{linesfallback=false}%
}
```

values based on 10pt vs 12pt

`ql@defaults@eqnlines` The (default) preset `eqnlines` implements a layout that scales with the font size by using the units `em` and `\normalbaselineskip` for horizontal and vertical spacing, respectively. It aims to approximately reproduce the `classic` spacing for a 12 pt computer modern font such that 10 pt fonts will lead to slightly reduced spacing. Apart from that, the `eqnlines` setting makes some deliberate layout choices that deviate significantly from `classic` (maximum column separation, no shortening below equations):

```
\def\eql@defaults@eqnlines{%
  \eqnlinesset{numberline=all}%
  \eqnlinesset{mintagsep=.5em}%
  \eqnlinesset{maxcolsep=2em}%
  \eqnlinesset{spread={0.2\normalbaselineskip}}%
  \eqnlinesset{tagmargin}%
  \eqnlinesset{tagmarginratio=.334}%
  \eqnlinesset{tagmarginthreshold=0.5}%
  \eqnlinesset{leftmargin={\leftmargini}}%
  \eqnlinesset{padding=0pt}%
  \eqnlinesset{bestlineauto}%
  \eqnlinesset{displayheight=strut}%
  \eqnlinesset{displaydepth=strut}%
  \eqnlinesset{shortmode=above}%
  \eqnlinesset{abovecontmode=noskip}%
  \eqnlinesset{belowcontmode=long}%
  \eqnlinesset{aboveparmode=long}%
  \eqnlinesset{belowparmode=long}%
  \eqnlinesset{abovetopmode=noskip}%
  \eqnlinesset{belowtopmode=long}%
  \eqnlinesset{longskip={0.75\normalbaselineskip
    plus 0.25\normalbaselineskip minus 0.4\normalbaselineskip}}%
  \eqnlinesset{aboveshortskip={0.0\normalbaselineskip
    plus 0.25\normalbaselineskip}}%
  \eqnlinesset{belowshortskip={0.0\normalbaselineskip
    plus 0.25\normalbaselineskip}}%
  \eqnlinesset{medskip={0.4\normalbaselineskip
    plus 0.2\normalbaselineskip minus 0.2\normalbaselineskip}}%
  \eqnlinesset{abovecontskip=0pt}%
  \eqnlinesset{belowcontskip=0pt}%
  \eqnlinesset{aboveparskip=0pt}%
  \eqnlinesset{belowparskip=0pt}%
  \eqnlinesset{abovetopskip=0pt}%
  \eqnlinesset{belowtopskip=0pt}%
  \eqnlinesset{abovetagskip=0pt}%
  \eqnlinesset{belowtagskip=0pt}%
  \eqnlinesset{abovemedtagskip=0pt}%
  \eqnlinesset{belowmedtagskip=0pt}%
  \eqnlinesset{abovepartagskip=0pt}%
  \eqnlinesset{belowpartagskip=0pt}%
  \eqnlinesset{crerror=true}%
  \eqnlinesset{linesfallback=true}%
}
```

P.5 Component Selection

The following routines provide several additional math environments beyond `equations`. They also backup and overwrite the original routines of LaTeX and amsmath carefully.

Tools.

\eql@provide@movecmd \eql@provide@moveenv eql@provide@movestar @provide@undefinecmd @provide@undefineenv

We introduce a couple of tools to rename and undefine commands and environments:

```
\def\eql@provide@movecmd#1#2{%
  \expandafter\let\csname #1\expandafter\endcsname\csname #2\endcsname
}
\def\eql@provide@moveenv#1#2{%
  \eql@provide@movecmd{#1}{#2}%
  \ifcsname end#2\endcsname
    \eql@provide@movecmd{end#1}{end#2}%
  \fi
}
\def\eql@provide@movestar#1#2{%
  \eql@provide@moveenv{#1}{#2}%
  \ifcsname #2*\endcsname
    \eql@provide@moveenv{#1*}{#2*}%
  \fi
}
\def\eql@provide@undefinecmd#1{%
  \expandafter\let\csname #1\endcsname\@undefined
}
\def\eql@provide@undefineenv#1{%
  \eql@provide@undefinecmd{#1}%
  \eql@provide@undefinecmd{end#1}%
}
```

Fix Endings for amsmath Environments. The amsmath derived environments forward their ending routines directly to the ending routines for the main environments `gather`, `multline`, `align`, `aligned`. This causes a problem when the main environments are replaced but the derived ones are still used. We fix the potential problem by copying the ending routines of the main environments to the ending routines of the derived environments.

\eql@amsmath@endfix

Check whether the original forwarding of an ending routine is still in place (other packages or future updates to amsmath might change the behaviour). If so, copy the ending routine into place:

```
\def\eql@amsmath@endfix#1#2{%
  \long\edef\@tempa{\expandafter\noexpand\csname end#2\endcsname}%
  \expandafter\ifx\csname end#1\endcsname\@tempa
    \eql@provide@movecmd{end#1}{end#2}%
  \fi
}
```

\eql@amsmath@fixends Perform the replacement for all amsmath environments whenever amsmath is loaded:

```
\def\eql@amsmath@fixends{%
  \eql@amsmath@after{%
    \eql@amsmath@endfix{gather*}{gather}%
    \eql@amsmath@endfix{multline*}{multline}%
    \eql@amsmath@endfix{align*}{align}%
    \eql@amsmath@endfix{flalign}{align}%
    \eql@amsmath@endfix{flalign*}{align}%
    \eql@amsmath@endfix{alignat}{align}%
    \eql@amsmath@endfix{alignat*}{align}%
    \eql@amsmath@endfix{xalignat}{align}%
    \eql@amsmath@endfix{xalignat*}{align}%
    \eql@amsmath@endfix{xxalignat}{align}%
    \eql@amsmath@endfix{gathered}{aligned}%
    \eql@amsmath@endfix{alignedat}{aligned}%
  }
}
```

Backup amsmath Environments. We can backup all amsmath environments *env* to `ams`*env* so that they can be used in parallel if needed.

ovide@backup@amsmath Copy an amsmath environment *env* to `ams`*env* whenever amsmath is loaded:

```
\def\eql@provide@backup@amsmath#1{%
  \eql@amsmath@after{%
    \eql@provide@moveenv{ams#1}{#1}%
  }%
}
```

provide@backup@eqref Copy an `eqref` to `amseqref` whenever amsmath is loaded:

```
\def\eql@provide@backup@eqref{%
  \eql@amsmath@after{%
    \eql@provide@movecmd{amseqref}{eqref}%
  }%
}
```

ide@backup@multlined The environment `multlined` is supplied by mathtools. We copy it to `amsmultlined` anyway, but whenever mathtools is loaded:

```
\def\eql@provide@backup@multlined{%
  \AddToHook{package/mathtools/after}{%
    \eql@provide@moveenv{amsmultlined}{multlined}%
  }%
}
```

vide@backup@equation The LaTeX environment `equation` is overwritten by several packages to implement their adjustments. Here we cater for adjustments through amsmath, hyperref and the pdf tagging mechanism. Copy `equation` and `equation*` whenever amsmath is loaded. Whenever hyperref is loaded, and amsmath is not yet present, backup the original LaTeX and hyperref versions of `equation`. If neither hyperref nor amsmath are present, just backup the original LaTeX `equation`. The pdf tagging mechanism registers `equation` upon `\begin{document}`. We thus need to register all copies of `equation` on our own, so that they can be used with their new names:

```
\def\eql@provide@backup@equation{%
  \eql@amsmath@after{%
```

```
    \eql@provide@moveenv{amsequation}{equation}%
    \eql@tagging@register@env{amsequation}%
    \eql@provide@moveenv{amsequation*}{equation*}%
    \eql@tagging@register@env{amsequation*}%
  }%
  \AddToHook{package/hyperref/after}{%
    \@ifpackageloaded{amsmath}{}{%
      \let\latexequation\H@equation
      \let\endlatexequation\H@endequation
      \eql@tagging@register@env{latexequation}%
      \eql@provide@moveenv{hyperrefequation}{equation}%
      \eql@tagging@register@env{hyperrefequation}%
    }%
  }%
  \@ifpackageloaded{amsmath}{}{\@ifpackageloaded{hyperref}{}{%
    \eql@provide@moveenv{latexequation}{equation}%
    \eql@tagging@register@env{latexequation}%
  }}%
}
```

`e@backup@displaymath` **TODO:** describe

```
\def\eql@provide@backup@displaymath{%
  \eql@provide@moveenv{latexdisplaymath}{displaymath}%
}
```

`@backup@subequations` The amsmath subequations environment is adjusted by hyperref through an environment hook, but this hook gets applied only later at `\begin{document}`. Hence, we need to supply the hook routine to the new routine ourselves:

```
\def\eql@provide@backup@subequations{%
  \eql@amsmath@after{%
    \eql@provide@moveenv{amssubequations}{subequations}%
  }%
  \AddToHook{package/hyperref/after}{%
    \AddToHook{cmd/amssubequations/before}%
    {%
      \stepcounter{equation}%
      \protected@edef\theHparentequation{\theHequation}%
      \addtocounter{equation}{-1}%
    }%
    \AddToHook{cmd/amssubequations/after}%
    {%
      \def\theHequation{\theHparentequation\alph{equation}}%
      \ignorespaces
    }%
  }%
}
```

`\eql@provide@backup` Backup all amsmath environments:

```
\def\eql@provide@backup{%
  \eql@provide@backup@eqref
  \eql@provide@backup@equation
  \eql@provide@backup@displaymath
  \eql@provide@backup@amsmath{gather}%
  \eql@provide@backup@amsmath{gather*}%
  \eql@provide@backup@amsmath{multline}%
  \eql@provide@backup@amsmath{multline*}%
  \eql@provide@backup@amsmath{align}%
  \eql@provide@backup@amsmath{align*}%
  \eql@provide@backup@amsmath{flalign}%
  \eql@provide@backup@amsmath{flalign*}%
  \eql@provide@backup@amsmath{alignat}%
  \eql@provide@backup@amsmath{alignat*}%
  \eql@provide@backup@amsmath{xalignat}%
  \eql@provide@backup@amsmath{xalignat*}%
  \eql@provide@backup@amsmath{xxalignat}%
  \eql@provide@backup@amsmath{aligned}%
  \eql@provide@backup@amsmath{aligned*}%
  \eql@provide@backup@amsmath{alignedat}%
  \eql@provide@backup@amsmath{alignedat*}%
  \eql@provide@backup@amsmath{gathered}%
  \eql@provide@backup@amsmath{gathered*}%
  \eql@provide@backup@multlined
  \eql@provide@backup@subequations
}
```

Replacement amsmath Environments.

`eql@gathered` (*env.*)
`eql@multlined` (*env.*)
`eql@aligned` (*env.*)

Define replacement versions for boxed environments `gathered`, `multlined` and `aligned` which forward to `equationsbox` with specific presets:

```
\newenvironment{eql@gathered}
  {\eqnaddopt{lines}\equationsbox}{\endequationsbox}
\newenvironment{eql@multlined}
  {\eqnaddopt{lines,padding,shape=steps}\equationsbox}{\endequationsbox}
\newenvironment{eql@aligned}
  {\eqnaddopt{align}\equationsbox}{\endequationsbox}
\newenvironment{eql@alignedat}[1]
  {\eqnaddopt{align,colsep=0pt}\equationsbox}{\endequationsbox}
```

`eql@equation` (*env.*)
`eql@gather` (*env.*)
`eql@multline` (*env.*)
`eql@align` (*env.*)

Define replacement versions for display environments `equation`, `gather`, `multline`, `aligned` and derivatives which forward to `equations` with specific presets: **TODO:** amsmath at variants would need predefined columns for full operation

```
\newenvironment{eql@equation}
  {\eqnaddopt{equation}\equations}{\endequations}
\newenvironment{eql@displaymath}
  {\eqnaddopt{equation,nonumber}\equations}{\endequations}
\newenvironment{eql@gather}
  {\eqnaddopt{lines}\equations}{\endequations}
\newenvironment{eql@multline}
  {\eqnaddopt{lines,padding=max,shape=steps,numberline=out}\equations}
  {\endequations}
\newenvironment{eql@align}
  {\eqnaddopt{align}\equations}{\endequations}
\newenvironment{eql@flalign}
  {\eqnaddopt{align,fulllength}\equations}{\endequations}
\newenvironment{eql@alignat}
  {\eqnaddopt{mincolsep=0pt,maxcolsep=0pt}\eql@xalignat}{\endequations}
\newenvironment{eql@xalignat}[1]
  {\eql@align}{\endequations}
\newenvironment{eql@xxalignat}[1]
  {\eql@flalign}{\endequations}
\newenvironment{eql@equation*}
  {\eqnaddopt{nonumber}\eql@equation}{\endequations}
```

```
\newenvironment{eql@gather*}
  {\eqnaddopt{nonumber}\eql@gather}{\endequations}
\newenvironment{eql@multline*}
  {\eqnaddopt{nonumber}\eql@multline}{\endequations}
\newenvironment{eql@align*}
  {\eqnaddopt{nonumber}\eql@align}{\endequations}
\newenvironment{eql@flalign*}
  {\eqnaddopt{nonumber}\eql@flalign}{\endequations}
\newenvironment{eql@alignat*}
  {\eqnaddopt{nonumber}\eql@alignat}{\endequations}
\newenvironment{eql@xalignat*}
  {\eqnaddopt{nonumber}\eql@xalignat}{\endequations}
```

Install Additional Environments. The additional environments need to be installed at their intended names which can be adjusted by the user.

`eql@provide@onlyonce` Process arguments for providing a specific environment. `#1` describes the environment using the amsmath name. `#2` specifies the desired target name. If `#2` is empty or equals `#1`, overwrite the amsmath environment in place making sure that the replacement is robust against loading amsmath before or after. If `#2` equals '`*`', just overwrite the amsmath environment in place immediately (e.g. within a block in the document body):

```
\def\eql@provide@onlyonce#1#2{%
  \def\eql@tmp{#2}%
  \def\@tempa{#1}%
  \ifx\eql@tmp\@tempa
    \let\eql@tmp\@empty
  \fi
  \ifx\eql@tmp\@empty
    \let\eql@tmp\@undefined
    \ifx\@nodocument\relax
      \def\eql@tmp{#1}%
    \fi
    \ifcsname eql@provided@#1\endcsname
      \def\eql@tmp{#1}%
    \fi
    \expandafter\let\csname eql@provided@#1\endcsname\eql@true
  \else
    \def\@tempa{*}%
    \ifx\eql@tmp\@tempa
      \def\eql@tmp{#1}%
    \fi
  \fi
}
```

`\eql@provide@eqref` Provide `\eqref` as the macro `#1`. We have to check whether `#1` is empty or equals `\eqref` or takes the value '`*`'. If not, we should strip the backslash for further processing. Copy the macro into place, and copy again when amsmath or mathtools are loaded. Remove definition before amsmath is loaded in the future to avoid a potential error:

```
\def\eql@provide@eqref#1{%
  \def\eql@tmp{#1}%
  \def\@tempa{\eqref}%
  \ifx\eql@tmp\@tempa
    \let\eql@tmp\@empty
  \fi
  \ifx\eql@tmp\@empty
```

```
    \eql@provide@onlyonce{eqref}{}%
  \else
    \def\@tempa{*}%
    \ifx\eql@tmp\@tempa
      \def\eql@tmp{eqref}%
    \else
      \edef\eql@tmp{\expandafter\@gobble\string#1}%
    \fi
  \fi
  \ifdefined\eql@tmp
    \expandafter\eql@provide@movecmd\expandafter{\eql@tmp}{eql@eqref}%
  \else
    \eql@amsmath@after{%
      \eql@provide@movecmd{eqref}{eql@eqref}%
    }%
    \AddToHook{package/mathtools/after}{%
      \eql@provide@movecmd{eqref}{eql@eqref}%
    }%
    \eql@provide@movecmd{eqref}{eql@eqref}%
    \eql@amsmath@undefine\eqref
  \fi
}
```

`\eql@provide@amsmath` Provide one of the amsmath environments and its star variant. Copy into place, and copy again when amsmath or mathtools are loaded. Remove definition before amsmath is loaded in the future to avoid an error:

```
\def\eql@provide@amsmath#1#2{%
  \eql@provide@onlyonce{#1}{#2}%
  \ifdefined\eql@tmp
    \expandafter\eql@provide@movestar\expandafter{\eql@tmp}{eql@#1}%
  \else
    \eql@amsmath@after{%
      \eql@provide@movestar{#1}{eql@#1}%
    }%
    \AddToHook{package/mathtools/after}{%
      \eql@provide@movestar{#1}{eql@#1}%
    }%
    \eql@provide@movestar{#1}{eql@#1}%
    \eql@amsmath@before{\eql@provide@undefineenv{#1}}%
    \ifcsname eql@#1*\endcsname
      \eql@amsmath@before{\eql@provide@undefineenv{#1*}}%
    \fi
  \fi
}
```

`ql@provide@multlined` Provide mathtools environment `multlined`. Copy into place, and copy again when amsmath or mathtools are loaded. Remove definition before mathtools is loaded in the future to avoid an error:

```
\def\eql@provide@multlined#1{%
  \eql@provide@onlyonce{multlined}{#1}%
  \ifdefined\eql@tmp
    \expandafter\eql@provide@moveenv\expandafter{\eql@tmp}{eql@multlined}%
  \else
    \AddToHook{package/mathtools/after}{%
      \eql@provide@moveenv{multlined}{eql@multlined}%
    }%
    \eql@provide@moveenv{multlined}{eql@multlined}%
    \@ifpackageloaded{mathtools}{}{\AddToHook{package/mathtools/before}{%
      \eql@provide@undefineenv{multlined}}}%
  \fi
}
```

eql@provide@equation Provide the environment `equation` and its star variant. Copy into place, and copy again when amsmath or hyperref are loaded. Remove definition of `equation*` before amsmath is loaded in the future to avoid an error. When pdf tagging is active, the environment is modified at `\begin{document}` in an undesirable fashion, so copy the definition again:

```
\def\eql@provide@equation#1{%
  \eql@provide@onlyonce{equation}{#1}%
  \ifdefined\eql@tmp
    \expandafter\eql@provide@movestar\expandafter{\eql@tmp}{eql@equation}%
  \else
    \eql@amsmath@after{%
      \eql@provide@movestar{equation}{eql@equation}%
    }%
    \AddToHook{package/hyperref/after}{%
      \@ifpackageloaded{amsmath}{}{%
        \eql@provide@moveenv{equation}{eql@equation}%
      }%
    }%
    \eql@provide@movestar{equation}{eql@equation}%
    \eql@amsmath@before{\eql@provide@undefineenv{equation*}}%
    \ifdefined\eql@tagging@on
      \AddToHook{begindocument/end}{%
        \eql@provide@movestar{equation}{eql@equation}%
      }%
    \fi
  \fi
}
```

@provide@displaymath **TODO:** describe

```
\def\eql@provide@displaymath#1{%
  \eql@provide@onlyonce{displaymath}{#1}%
  \ifdefined\eql@tmp
    \expandafter\eql@provide@moveenv\expandafter{\eql@tmp}{eql@displaymath}%
  \else
    \eql@provide@moveenv{displaymath}{eql@displaymath}%
    \ifdefined\eql@tagging@on
      \AddToHook{begindocument/end}{%
        \eql@provide@moveenv{displaymath}{eql@displaymath}%
      }%
    \fi
  \fi
}
```

provide@subeqiations Provide the amsmath environment `subequations`. Copy into place, and copy again when amsmath is loaded. hyperref adds a hook to the command which messes up the parsing of optional arguments (even if the hook is emptied). The hook placement happens at `\begin{document}`, so we copy the environment again afterwards. We also remove the hook (after adding an empty hook to avoid errors). Remove definition before amsmath is loaded in the future to avoid an error:

```
\def\eql@provide@subequations#1{%
  \eql@provide@onlyonce{subequations}{#1}%
  \ifdefined\eql@tmp
    \expandafter\eql@provide@moveenv
      \expandafter{\eql@tmp}{eql@subequations}%
  \else
    \eql@amsmath@after{%
      \eql@provide@moveenv{subequations}{eql@subequations}%
    }%
    \AddToHook{package/hyperref/after}{%
      \AddToHook{cmd/subequations/before}[hyperref]{}%
      \AddToHook{cmd/subequations/after}[hyperref]{}%
      \RemoveFromHook{cmd/subequations/before}[hyperref]%
      \RemoveFromHook{cmd/subequations/after}[hyperref]%
      \AddToHook{begindocument/end}{%
        \eql@provide@moveenv{subequations}{eql@subequations}%
      }%
    }%
    \eql@provide@moveenv{subequations}{eql@subequations}%
    \eql@amsmath@before{\eql@provide@undefineenv{subequations}}%
  \fi
}
```

`\eql@provide@sqr` Provide the symbolic environment `\[...\]`. Copy into place, and copy again when amsmath is loaded. If pdf tagging is active, some undesired modifications happen at `\begin{document}`, so copy again afterwards:

```
\def\eql@provide@sqr{%
  \let\[\eql@sqr@open
  \let\]\eql@sqr@close
  \eql@amsmath@after{%
    \let\[\eql@sqr@open
    \let\]\eql@sqr@close
  }%
  \ifdefined\eql@tagging@on
    \AddToHook{begindocument/end}{%
      \let\[\eql@sqr@open
      \let\]\eql@sqr@close
    }%
  \fi
}
```

`\eql@provide@ang` Provide the symbolic environment `\<...\>`. This is easy because none of the other packages uses this structure:

```
\def\eql@provide@ang{%
  \let\<\eql@ang@open
  \let\>\eql@ang@close
}
```

Interface.

`provide` (*key*) We provide the additional environments via key-value pairs, where the value specifies the intended name:

```
\eql@define@key{provide}{equation}[]{\eql@provide@equation{#1}}
\eql@define@key{provide}{displaymath}[]{\eql@provide@displaymath{#1}}
\eql@define@key{provide}{gather}[]{\eql@provide@amsmath{gather}{#1}}
\eql@define@key{provide}{multline}[]{\eql@provide@amsmath{multline}{#1}}
\eql@define@key{provide}{align}[]{\eql@provide@amsmath{align}{#1}}
```

```
\eql@define@key{provide}{flalign}[]{\eql@provide@amsmath{flalign}{#1}}
\eql@define@key{provide}{alignat}[]{\eql@provide@amsmath{alignat}{#1}}
\eql@define@key{provide}{xalignat}[]{\eql@provide@amsmath{xalignat}{#1}}
\eql@define@key{provide}{xxalignat}[]{\eql@provide@amsmath{xxalignat}{#1}}
\eql@define@key{provide}{aligned}[]{\eql@provide@amsmath{aligned}{#1}}
\eql@define@key{provide}{alignedat}[]{\eql@provide@amsmath{alignedat}{#1}}
\eql@define@key{provide}{gathered}[]{\eql@provide@amsmath{gathered}{#1}}
\eql@define@key{provide}{multlined}[]{\eql@provide@multlined{#1}}
\eql@define@key{provide}{subequations}[]{\eql@provide@subequations{#1}}
\eql@define@key{provide}{sqr}[]{\eql@provide@sqr}
\eql@define@key{provide}{ang}[]{\eql@provide@ang}
\eql@define@key{provide}{eqref}[]{\eql@provide@eqref{#1}}
\eql@define@key{provide}{tagform}[]{%
  \def\tagform@##1{\maketag@@@{\eql@tag@form{#1}}}}
\eql@define@key{provide}{maketag}[]{%
  \def\maketag@@@##1{\eql@tag@box{##1}}}
```

`\eqnlinesprovide` Provide an additional environment or macro via key-value interface:

```
\newcommand{\eqnlinesprovide}[1]{%
⟨dev⟩\eql@dev@start\eqnlinesprovide
  \eql@setkeys{provide}{#1}%
  \ignorespaces
}
```

P.6 Global and Package Options

Handle global and package options:

`\eqnlinesset` The macro `\eqnlinesset` processes global configuration options including the package options:

```
\newcommand{\eqnlinesset}[1]{%
⟨dev⟩\eql@dev@start\eqnlinesset
  \eql@setkeys{setup}{#1}%
  \ignorespaces
}
```

Disable error message for exclusive package options:

```
\let\eql@error@packageoption\@gobble
```

Declare math layout options `leqno` and `fleqn` for common LaTeX classes:

```
\DeclareOption{leqno}{\eqnlinesset{tagsleft}}
\DeclareOption{fleqn}{\eqnlinesset{left}}
```

Pass undeclared options on to keyval processing:

```
\DeclareOption*{\expandafter\eqnlinesset\expandafter{\CurrentOption}}
```

Set defaults for package:

```
\eql@defaults@eqnlines
\eql@mode@columns
\eql@mode@aligned
```

Process package options:

```
\ProcessOptions
```

@error@packageoption Enable error message for exclusive package options:

```
\def\eql@error@packageoption#1{%
  \eql@error{may only use ‘#1’ as a package option}%
}
```

Make sure that the amsmath conditionals `\iftagsleft@` and `\if@fleqn` are declared without spelling out their name which may upset the TEX conditional parsing mechanism:

```
\ifdefined\tagsleft@true\else
  \expandafter\newif\csname iftagsleft@\endcsname
\fi
\ifdefined\@fleqntrue\else
  \expandafter\newif\csname if@fleqn\endcsname
\fi
```

Import amsmath switches `leqno` as `tagsleft` and `fleqn` as `left`:

```
\ifdefined\eql@provide@opt@amsmath
  \let\eql@provide@opt@equation\eql@true
  \eql@amsmath@after{%
    \iftagsleft@
      \eqnlinesset{tagsleft}
    \else
      \eqnlinesset{tagsright}
    \fi
    \if@fleqn
      \eqnlinesset{left}
    \else
      \eqnlinesset{center}
    \fi
  }
\fi
```

Make the ending statements for amsmath environments independent if desired, so that they may be overwritten individually:

```
\ifdefined\eql@provide@opt@amsmathends\eql@amsmath@fixends\fi
```

Backup all amsmath environments that may be overwritten to `ams...`. This will happen before any replacements:

```
\ifdefined\eql@provide@opt@backup\eql@provide@backup\fi
```

Provide native LATEX environment `equation` and symbolic shortcut `\[...\]` if desired:

```
\ifdefined\eql@provide@opt@equation\eqnlinesprovide{%
  equation,sqr,displaymath}
\fi
```

Provide amsmath equation environments if desired:

```
\ifdefined\eql@provide@opt@amsmath
  \eqnlinesprovide{%
    multline,gather,align,flalign,alignat,xalignat,xxalignat,%
    multlined,gathered,aligned,alignedat,%
    subequations}
\fi
```

Provide symbolic shortcut `\<...\>` if desired:

```
\ifdefined\eql@provide@opt@ang\eqnlinesprovide{ang}\fi
```

Provide equation reference `\eqref` if desired:

```
\ifdefined\eql@provide@opt@eqref\eqnlinesprovide{eqref}\fi
```