The `equationarray` environment

Roland Winkler
rwinkler@niu.edu

December 1, 2010 v1.3

Abstract

This package provides the `equationarray` environment. It combines the line numbering of the `eqnarray` and the more flexible formatting features of the `array` environment. It requires the `array` package.

1 Introduction

Problem: The `eqnarray` environment is too restrictive because it is only prepared to accept three part equations. Some applications require more sophisticated mathematics, so three parts aren't enough. Simultaneously, we would like to have our equations numbered. If the equations aren't to be numbered, then the `array` environment suffices.[1] If the equations aren't to be aligned with each other, then the `equation` environment is preferred.

A first version of the `equationarray` environment was written by Tony Li, University of Southern California, `<tli@sargas.usc.edu>` starting 6/15/1988. The current version has been completely rewritten in order to make it compatible with Frank Mittelbach's `array` environment, i.e., it should be possible to use all features of the `array` environment. If you find a bug (see below!) or if you make any improvements, I'd like to hear about them.

2 Example

We give a short example for the use of `equationarray`. The text is

```
\newcolumntype{e}{@{\quad}}
\arraycolsep 0.2em
\begin{equationarray}{p{2.5em}erclecercl}
```

[1] Nevertheless it is often easier to use the `equationarray*` environment than both an `equation` plus an `array` environment. Moreover a page break is possible between two lines of the `equationarray`.

```
now: & - i\partial_t \psi & = & H\,\psi & \Rightarrow
& \psi (t) & = & \psi (0) \exp (iEt) \\[3ex]
then: & - i\partial_t \psi & = & (H+E_0) \,\psi & \Rightarrow
& \psi (t) & = & \psi (0) \exp [i(E+E_0)t]
\end{equationarray}
```

and we obtain

$$\text{now:} \quad -i\partial_t\psi = H\,\psi \quad \Rightarrow \quad \psi(t) = \psi(0)\exp(iEt) \tag{1}$$

$$\text{then:} \quad -i\partial_t\psi = (H+E_0)\,\psi \quad \Rightarrow \quad \psi(t) = \psi(0)\exp[i(E+E_0)t] \tag{2}$$

An `equationarray` behaves very much like an `array`. For example, `equationarray` accepts the same tokens for defining columns, and new column types can be defined with `\newcolumntype`. One can use `\multicolumn`, `\hline`, `\cline`, and `\vline`. Moreover, `equationarray` makes use of `\arraycolsep`, `\extracolsep`, and `\extrarowheight`.[2] In an `equationarray` all these things have the same meaning like in an `array`. The main difference is that by default each entry of an `equationarray` is displayed with `\displaystyle` whereas `array` uses `\textstyle`. One can avoid page breaks between two lines by using the star version `\\*`. There is also the star version `equationarray*` which has usually no line numbers. But if for a particular line you still want a line number, you can use `\yesnumber`. (I find this more convenient than having many lines with `\nonumber`.) Inside the `equationarray` and `equationarray*` environments, there is also the macro `\eqnnum` available. It takes one argument that is used as the equation number for the current line. The normal line number is always suppressed in such a line.

The default is that without the `fleqn` option the equations are centered. With the `fleqn` option they are left-justified, indented by `\mathindent`.[3] With the optional argument `[l]` or `[r]` the `equationarray` will appear flushleft or flushright, with the option `[c]` the `equationarray` will be centered, e.g. `\begin{equationarray}[l]{rclll}`.

3 Bugs

Compared with `array` the `equationarray` uses slightly extended versions of `\@classz` and `\multicolumn` because it must count the columns. Each time you insert a `&` the counter `\@eqcnt` is increased by one. `\multicolumn` increases `\@eqcnt` according to the number of `\span`ed columns. Thus if you

[2] Some people don't like the large spacing between the columns of the standard `eqnarray`. Don't be surprised that `equationarray` seems to have the same "bug". To decrease the spacing between two columns you just have to change the value of `\arraycolsep`, see the example above.

[3] Thanks to Piet van Oostrum `<piet@cs.ruu.nl>` who added the code which is necessary for the `fleqn` option.

have an `array` within the `equationarray` environment and in that line of the `equationarray` you have less `&`'s than columns defined in the preamble, then the line numbering will be not flushright but further to the left.

In that case you can either fill up the end of the line with extra `&`'s (straightforward) or you can put the original definition of `\@classz` and `\multicolumn` within the definition of the `array` command.

4 License

This LaTeXpackage is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This package is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

The GNU General Public License is available at
`http://www.gnu.org/licenses/gpl.html`

5 The code

```
\typeout{equationarray \fileversion\space<\filedate>}
% \typeout{English documentation\space\space<\docdate>}
\NeedsTeXFormat{LaTeX2e} \ProvidesPackage{eqnarray}
```

`equationarray` can't do anything if we don't have the `array` package.

```
\RequirePackage{array}
```

Process the `fleqn` option.

```
\def\eqnarr@left{\@centering}
\let\eqnarr@opts\relax
\DeclareOption{fleqn}{
  \def\eqnarr@left{\mathindent}
  \def\eqnarr@opts{\displaywidth\linewidth
                   \advance\displaywidth-\mathindent} }
\ProcessOptions
```

`\equationarray`

```
\def\equationarray{%
 \col@sep\arraycolsep
 \def\d@llarbegin{$\displaystyle}%
 \def\d@llarend{$}%
 \stepcounter{equation}%
 \let\@currentlabel=\theequation
 \set@eqnsw \global\@eqcnt\z@ \global\@eqargcnt\z@
 \let\@classz\@eqnclassz
```

We need an extended definition of `\multicolumn` which increases the counter `\@eqcnt` according to the number of columns covered by `\multicolumn`.

```
\def\multicolumn##1##2##3{\@eqnmulticolumn{##1}{##2}{##3}%
                          \global\advance\@eqcnt##1
                          \global\advance\@eqcnt\m@ne}%
```

The macro `\eqnnum` initializes `\@eqnarrnum` and suppresses ordinary equation numbering.

```
\def\eqnnum##1{\global\@eqnswfalse\gdef\@eqnarrnum{##1}}
\def\@halignto{to\displaywidth}%
\@ifnextchar[{\@equationarray}{\@equationarray[.]}}
```

`\@eqnmulticolumn` `\@eqnmulticolumn` equals the original version of `\multicolumn`.

```
\let\@eqnmulticolumn=\multicolumn
```

`\nonumber,\yesnumber` Note that `\nonumber` is already defined in standard latex.tex

```
% \def\nonumber{\global\@eqnswfalse}
\def\yesnumber{\global\@eqnswtrue}
\let\set@eqnsw=\yesnumber
```

`\@amper` We need a macro for `&` that expands at the right time.

```
\def\@amper{&}
```

`\@eqargcnt` The main idea about the `equationarray` is the following: The counter `\@eqargcnt` counts the number of columns defined in the preamble. In each line of the `equationarray`, the counter `\@eqcnt` counts the number of `&` which have been introduced explicitely by the user. By comparing these counters we can silently fill every line of the `equationarray` with exactly `\@eqargcnt` copies of `&` before we insert the equation number. See also the TEXbook, Exercise 22.9.

```
\newcount\@eqargcnt  % counts number of columns
```

`\@equationarray`

```
\def\@equationarray[#1]#2{%
     \eqnarr@opts
     \@tempdima \ht \strutbox
     \advance \@tempdima by\extrarowheight
     \setbox\@arstrutbox=\hbox{\vrule
               \@height\arraystretch \@tempdima
               \@depth\arraystretch \dp \strutbox
               \@width\z@}%
     \gdef\advance@eqargcnt{\global\advance\@eqargcnt\@ne}%
     \begingroup
     \@mkpream{#2}%
     \xdef\@preamble{%
      \if #1l\tabskip\z@ \else\if #1r\tabskip\@centering
                         \else\if #1c\tabskip\@centering
                         \else\tabskip\eqnarr@left \fi\fi\fi
      \halign \@halignto
      \bgroup \tabskip\z@ \@arstrut \@preamble
```

```
      \if #1l\tabskip\@centering \else\if #1r\tabskip\z@
                                  \else\tabskip\@centering \fi\fi
```

Here we need an extra column for the equation-number

```
      \@amper\llap{\@sharp}\tabskip\z@\cr}%
    \endgroup
    \gdef\advance@eqargcnt{}%
    \bgroup
    \let\@sharp## \let\protect\relax
    \m@th   \let\\=\@equationcr
    \let\par\@empty
    $$                            % $$ BRACE MATCHING HACK
    \lineskip \z@
    \baselineskip \z@
    \@preamble}
```

`\@eqnclassz` `\@eqnclassz` does the same thing as `\@classz` except that we add `\advance@eqargcnt`

```
  \def\@eqnclassz{\@classx
     \@tempcnta \count@
     \advance@eqargcnt
     \prepnext@tok
     \@addtopreamble{%
        \global\advance\@eqcnt\@ne
        \ifcase \@chnum
        \hfil \d@llarbegin \insert@column \d@llarend\hfil \or
        \d@llarbegin \insert@column \d@llarend \hfil \or
        \hfil\kern\z@ \d@llarbegin \insert@column \d@llarend \or
        $\vcenter
        \@startpbox{\@nextchar}\insert@column \@endpbox $\or
        \vtop \@startpbox{\@nextchar}\insert@column \@endpbox \or
        \vbox \@startpbox{\@nextchar}\insert@column \@endpbox
        \fi}\prepnext@tok}
```

`\endequationarray`

```
  \def\endequationarray{\@zequationcr
     \egroup
     \global\advance\c@equation\m@ne $$  % $$ BRACE MATCHING HACK
     \egroup\global\@ignoretrue
     \gdef\@preamble{}}
```

`\@equationcr` If we have `\\*` the command `\@equationcr` avoids page breaks

```
  \def\@equationcr{${\ifnum0=`}\fi\@ifstar{\global\@eqpen\@M
      \@xequationcr}{\global\@eqpen\interdisplaylinepenalty
                    \@xequationcr}}
```

`\@xequationcr`

```
  \def\@xequationcr{%
      \@ifnextchar[{\@argequationcr}{\ifnum0=`{\fi}${}%
      \@zequationcr}}
```

\@argequationcr

```
\def\@argequationcr[#1]{\ifnum0=‘{\fi}${}\ifdim #1>\z@
   \@xargequationcr{#1}\else
   \@yargequationcr{#1}\fi}
```

\@xargequationcr

```
\def\@xargequationcr#1{\unskip
   \@tempdima #1\advance\@tempdima \dp \@arstrutbox
   \vrule \@depth\@tempdima \@width\z@
   \@zequationcr\noalign{\penalty\@eqpen}}
```

\@yargequationcr

```
\def\@yargequationcr#1{%
   \@zequationcr\noalign{\penalty\@eqpen\vskip #1}}
```

\@eqnarrnum The macro \@eqnarrnum holds a customized equation number. This macro is initialized via the macro \eqnnum.

```
\let\@eqnarrnum\relax
```

\@zequationcr We add &\omit for those columns that will remain empty. Note that without \omit we already have \advance\@eqcnt\@ne in the preamble.

```
\def\@zequationcr{\@whilenum\@eqcnt <\@eqargcnt
   \do{\@amper\omit\global\advance\@eqcnt\@ne}%
```

We add an extra alignment tab for the equationnumber

```
   \@amper
```

Either we insert the regular equation number or the customized one.

```
   \if@eqnsw\@eqnnum\stepcounter{equation}\else
     \@eqnarrnum\global\let\@eqnarrnum\relax\fi
   \set@eqnsw\global\@eqcnt\z@\cr}
```

\equationarray* Finally we define the equationarray* environment. It does exactly the same thing as \equationarray except that we \let the command \set@eqnsw equal \nonumber

```
\@namedef{equationarray*}{%
   \let\set@eqnsw=\nonumber \equationarray}
\@namedef{endequationarray*}{\endequationarray}
```