

# HOW TO TYPESET EQUATIONS IN L<sup>A</sup>T<sub>E</sub>X

Stefan M. Moser

29 September 2009, Version 3.2

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Single Equations: <code>equation</code></b>	<b>2</b>
<b>3</b>	<b>Single Equations that are Too Long: <code>multline</code></b>	<b>3</b>
3.1	Case 1: The Expression Is Not an Equation . . . . .	4
3.2	Case 2: A Single Equation To Be Wrapped Before Equality Sign . .	5
3.3	Case 3: Additional Comment . . . . .	5
3.4	Case 4: LHS Too Long—RHS Too Short . . . . .	5
3.5	Case 5: A Term on RHS Should Not Be Split . . . . .	5
<b>4</b>	<b>Multiple Equations: <code>IEEEeqnarray</code></b>	<b>6</b>
4.1	Problems with Traditional Commands . . . . .	6
4.2	<code>IEEEeqnarray</code> . . . . .	8
4.3	Usage in Most Common Situations . . . . .	9
<b>5</b>	<b>Advanced Typesetting</b>	<b>12</b>
5.1	<code>IEEEeqnarraybox</code> : General Tables and Arrays . . . . .	12
5.2	Case-Distinctions . . . . .	14
5.3	Matrices . . . . .	15
5.4	Framed Equations . . . . .	16
5.5	More Fancy Frames . . . . .	18
5.6	Putting the QED Right . . . . .	19
<b>6</b>	<b>Emacs and <code>IEEEeqnarray</code></b>	<b>22</b>
<b>7</b>	<b>Some Final Remarks</b>	<b>23</b>

## 1 Introduction

L<sup>A</sup>T<sub>E</sub>X is a very powerful tool for typesetting in general and for typesetting math in particular. However, in spite of its power there are still many ways of generating better or less good results. This manual offers some tricks and hints that hopefully will lead to the former...

Note that this manual does neither claim to provide the best nor the only solution. Its aim is rather to give a couple of rules that can be followed easily and that will lead to a good layout of all equations in a document. It is assumed that the reader has already mastered the basics of L<sup>A</sup>T<sub>E</sub>X.

The structure of this document is as follows: we introduce the most basic equation in Section 2. Section 3 then explains some first possible reactions when an equation is too long. The probably most important part is contained in Section 4: there we introduce the extremely powerful `IEEEeqnarray`-environment that should be used in any case instead of `align` or `eqnarray`.

In Section 5 some more advanced problems and possible solutions are discussed, and Section 6 contains some hints and tricks about the editor Emacs.

In the following any L<sup>A</sup>T<sub>E</sub>X-command will be set in **typewriter font**. *RHS* stands for *right-hand side*, *i.e.*, all terms on the right of the equality (or inequality) sign. Similarly, *LHS* stands for *left-hand side*, *i.e.*, all terms on the left of the equality sign. To simplify language we will usually talk about *equality*. Obviously, the typesetting does not change if an expression actually is an *inequality*.

This documents comes together with some additional files that might be helpful:

- `typeset_equations.tex`: L<sup>A</sup>T<sub>E</sub>X-source code file of this manual;
- `dot_emacs`: commands to include in your `.emacs`-file (see Section 6);
- `IEEEtrantools.sty` [2007/01/11 V1.2 by Michael Shell]: package needed for the `IEEEeqnarray`-environment;
- `IEEEtran_HOWTO.pdf`: official manual of the `IEEEtran`-class. The part about `IEEEeqnarray` is found in Appendix F.

## 2 Single Equations: equation

The main strength of L<sup>A</sup>T<sub>E</sub>X concerning typesetting of mathematics is based on the package `amsmath`. Every current distribution of L<sup>A</sup>T<sub>E</sub>X will come with this package included, so you only need to make sure that you include the following line in the header of your document:

```
\usepackage{amsmath}
```

Throughout this document it is assumed that `amsmath` is loaded.

Single equations should be exclusively typed using the `equation`-environment:

```
\begin{equation}
  a = b + c
\end{equation}
```

$$a = b + c \quad (1)$$

In case one does not want to have an equation number, the `*`-version is used:

```
\begin{equation*}
  a = b + c
\end{equation*}
```

$$a = b + c$$

All other possibilities of typesetting simple equations have disadvantages:

- The `displaymath`-environment offers no equation numbering. To add or to remove a “\*” in the `equation`-environment is much more flexible.
- Commands like `$$...$$`, `\[...\]`, *etc.*, have the additional disadvantage that the source code is extremely poorly readable. Moreover, `$$...$$` is faulty as the vertical space after the equation is too large in certain situations.

We summarize:

- ▷ *For all the above mentioned reasons we should exclusively use `equation` (and no other environment) to produce a single equation.*

### 3 Single Equations that are Too Long: `multline`

If an equation is too long, we have to wrap it somehow. Unfortunately, wrapped equations are usually less easy to read than not-wrapped ones. To improve the readability, there are certain rules on how to do the wrapping:

1. In general one should always wrap an equation **before** an equality sign or an operator.
2. A wrap before an equality sign is preferable to a wrap before any operator.
3. A wrap before a plus- or minus-operator is preferable to a wrap before a multiplication-operator.
4. Any other type of wrap should be avoided if ever possible.

The easiest way to achieve such a wrapping is the use of the `multline`-environment:<sup>1</sup>

```
\begin{multline}
a + b + c + d + e + f
+ g + h + i
\\
= j + k + l + m + n
\end{multline}
```

$$a + b + c + d + e + f + g + h + i \\ = j + k + l + m + n \quad (2)$$

The difference to the `equation`-environment is that an arbitrary line-break (or also multiple line-breaks) can be introduced. This is done by putting a `\\` on those places where the equation needs to be wrapped. Similarly to `equation*` there also exists a `multline*`-version for preventing an equation number.

<sup>1</sup>As a reminder: it is necessary to include the `amsmath`-package for this command to work!

However, in spite of its ease in use, often the `IEEEeqnarray`-environment (see Section 4) will yield better results. Particularly, consider the following common situation:

```
\begin{equation}
  a = b + c + d + e + f
    + g + h + i + j
    + k + l + m + n + o
  \label{eq:equation_too_long}
\end{equation}
```

$$a = b + c + d + e + f + g + h + i + j + k + l + m + n + o \quad (3)$$

Here it is actually the RHS that is too long to fit on one line. The `multline`-environment will now yield the following:

```
\begin{multline}
  a = b + c + d + e + f
    + g + h + i + j \\
    + k + l + m + n + o
\end{multline}
```

$$a = b + c + d + e + f + g + h + i + j \\ + k + l + m + n + o \quad (4)$$

This is of course much better than (3), but it has the disadvantage that the equality sign loses its natural stronger importance with respect to the plus operator in front of  $k$ . The better solution is provided by the `IEEEeqnarray`-environment that will be discussed in detail in Section 4:

```
\begin{IEEEeqnarray}{rCl}
  a & = & b + c + d + e + f \\
    & + & g + h + i + j \nonumber \\
    & & k + l + m + n + o \\
  \label{eq:dont_use_multline}
\end{IEEEeqnarray}
```

$$a = b + c + d + e + f + g + h + i + j \\ + k + l + m + n + o \quad (5)$$

In this case the second line is vertically aligned to the first line: the  $+$  in front of  $k$  is exactly below  $b$ , *i.e.*, the RHS is clearly visible as contrast to the LHS of the equation.

▷ *The `multline`-environment should exclusively be used in the following five specific situations only.*

### 3.1 Case 1: The Expression Is Not an Equation

If the expression is not an equation, *i.e.*, there is no equality sign, then there exists no RHS or LHS and `multline` offers a nice solution:

```
\begin{multline}
  \lim_{n \uparrow \infty} \{ \\
  \Big\{ a + b + c + d + e + f \\
    + g + h + i + j + k + l \\
    + m + n + o + p + q \\
  \Big\} \\
\end{multline}
```

$$\lim_{n \uparrow \infty} \left\{ a + b + c + d + e + f \right. \\ \left. + g + h + i + j + k + l \right. \\ \left. + m + n + o + p + q \right\} \quad (6)$$

### 3.2 Case 2: A Single Equation To Be Wrapped Before Equality Sign

If a single equation is too long, but it can be wrapped in front of the equality sign, then this usually will yield a good solution where LHS and RHS are still clearly visible:

```
\begin{multline}
  a + b + c + d + e + f
  \\ = g + h + i + j
  + k + l + m
\end{multline}
```

$$\begin{aligned} a + b + c + d + e + f \\ = g + h + i + j + k + l + m \end{aligned} \quad (7)$$

### 3.3 Case 3: Additional Comment

If there is an additional comment at the end of the equation that does not fit on the same line, then this comment can be put onto the next line:

```
\begin{multline}
  a + b + c + d
  = e + f + g + h, \quad \text{\quad} \\
  \text{\textnormal{for } } 0 \leq n \\
  \text{\textnormal{le } } n_{\text{\textnormal{max}}} \\
\end{multline}
```

$$\begin{aligned} a + b + c + d = e + f + g + h, \\ \text{for } 0 \leq n \leq n_{\max} \end{aligned} \quad (8)$$

### 3.4 Case 4: LHS Too Long—RHS Too Short

If the LHS of a single equation is too long and the RHS is very short, then one cannot break the equation in front of the equality sign as wished, but one is forced to do it somewhere on the LHS. In this case one cannot nicely keep the natural separation of LHS and RHS anyway and `multline` offers the best (of bad) solutions:

```
\begin{multline}
  a + b + c + d + e + f
  + g \quad + h + i + j
  + k + l = m
\end{multline}
```

$$\begin{aligned} a + b + c + d + e + f + g \\ + h + i + j + k + l = m \end{aligned} \quad (9)$$

### 3.5 Case 5: A Term on RHS Should Not Be Split

The following is a special (and rather rare) case: the LHS would be short enough and/or the RHS long enough in order to wrap the equation in front of the equality sign. This usually would call for the `IEEEeqnarray`-environment, see (5). However, one term on the RHS is an entity that we rather would not split, but it is too long to fit:

```

\begin{multline}
h^{-}(\mathbf{X}|\mathbf{Y}) \leq \frac{n+1}{e} - h(\mathbf{X}|\mathbf{Y}) \\
+ \int p(\mathbf{y}) \log \left( \frac{\mathbb{E}[\|\mathbf{X}\|^2 | \mathbf{Y} = \mathbf{y}]}{n} \right) d\mathbf{y}
\end{multline}

```

$$\begin{aligned}
h^{-}(\mathbf{X}|\mathbf{Y}) &\leq \frac{n+1}{e} - h(\mathbf{X}|\mathbf{Y}) \\
&+ \int p(\mathbf{y}) \log \left( \frac{\mathbb{E}[\|\mathbf{X}\|^2 | \mathbf{Y} = \mathbf{y}]}{n} \right) d\mathbf{y}
\end{aligned} \tag{10}$$

In this example the integral on the RHS is too long, but should not be split for readability.

Note that even in this case it might be possible to find different, possibly better solutions based on `IEEEeqnarray`-environment:

```

\begin{IEEEeqnarray}{rCl}
\IEEEeqnarraymulticol{3}{1}{
h^{-}(\mathbf{X}|\mathbf{Y})
}\nonumber\\
&\leq \frac{n+1}{e} - h(\mathbf{X}|\mathbf{Y}) \\
&+ \int p(\mathbf{y}) \log \left( \frac{\mathbb{E}[\|\mathbf{X}\|^2 | \mathbf{Y} = \mathbf{y}]}{n} \right) d\mathbf{y}
\nonumber
\end{IEEEeqnarray}

```

$$\begin{aligned}
h^{-}(\mathbf{X}|\mathbf{Y}) \\
&\leq \frac{n+1}{e} - h(\mathbf{X}|\mathbf{Y}) \\
&+ \int p(\mathbf{y}) \log \left( \frac{\mathbb{E}[\|\mathbf{X}\|^2 | \mathbf{Y} = \mathbf{y}]}{n} \right) d\mathbf{y}
\end{aligned} \tag{11}$$

## 4 Multiple Equations: `IEEEeqnarray`

In the most general situation we have a sequence of several equalities that do not fit onto one line. Here we need to work with vertical alignment in order to keep the array of equations in a nice and readable structure.

Before we offer our suggestions on how to do this, we start with a few bad examples that show the biggest drawbacks of some common solutions.

### 4.1 Problems with Traditional Commands

To group multiple equations the `align`-environment<sup>2</sup> could be used:

<sup>2</sup>The `align`-environment can also be used to group several blocks of equations beside each other. However, for this rather rare situation we also recommend to use the `IEEEeqnarray`-environment with an argument like, *e.g.*, `{rCl+rCl}`.

```
\begin{align}
a &= b + c \\
&= d + e
\end{align}
```

$$a = b + c \quad (12)$$

$$= d + e \quad (13)$$

However, this approach does not work once a single line is too long:

```
\begin{align}
a &= b + c \\
&= d + e + f + g + h + i \\
&+ j + k + l \nonumber \\
&+ m + n + o \\
&= p + q + r + s
\end{align}
```

$$a = b + c \quad (14)$$

$$= d + e + f + g + h + i + j + k + l$$

$$+ m + n + o \quad (15)$$

$$= p + q + r + s \quad (16)$$

Here  $+m$  should be below  $d$  and not below the equality sign. Of course, one could add some space by, *e.g.*, `\hspace{...}`, but this will never yield a precise arrangement (and is bad style...).

A better solution is offered by the `eqnarray`-environment:

```
\begin{eqnarray}
a &= & b + c \\
&= & d + e + f + g + h + i \\
&+ j + k + l \nonumber \\
&&+ m + n + o \\
&= & p + q + r + s
\end{eqnarray}
```

$$a = b + c \quad (17)$$

$$= d + e + f + g + h + i + j + k + l$$

$$+ m + n + o \quad (18)$$

$$= p + q + r + s \quad (19)$$

The `eqnarray`-environment, however, has a few very severe disadvantages:

- The spaces around the equality signs are too big. Particularly, they are **not** the same as in the `multline`- and `equation`-environments:

```
\begin{eqnarray}
a &= & a = a
\end{eqnarray}
```

$$a = a = a \quad (20)$$

- The expression sometimes overlaps with the equation number even though there would be enough room on the left:

```
\begin{eqnarray}
a &= & b + c \\
&= & d + e + f + g + h^2 \\
&+ i^2 + j
\label{eq:faultyeqnarray}
\end{eqnarray}
```

$$a = b + c \quad (21)$$

$$= d + e + f + g + h^2 + i^2 + j \quad (22)$$

- The `eqnarray`-environment offers a command `\lefteqn{...}` that can be used when the LHS is too long:

```
\begin{eqnarray}
\lefteqn{a + b + c + d
+ e + f + g + h}\nonumber\\
&= & i + j + k + l + m \\
\\
&= & n + o + p + q + r + s
\end{eqnarray}
```

$$\begin{aligned} a + b + c + d + e + f + g + h \\ &= i + j + k + l + m & (23) \\ &= n + o + p + q + r + s & (24) \end{aligned}$$

Unfortunately, this command is faulty: if the RHS is too short, the array is not properly centered:

```
\begin{eqnarray}
\lefteqn{a + b + c + d
+ e + f + g + h}
\nonumber \\
&= & i + j \\
\end{eqnarray}
```

$$\begin{aligned} a + b + c + d + e + f + g + h \\ &= i + j & (25) \end{aligned}$$

Moreover, it is very complicated to change the vertical alignment of the equality sign on the second line.

To overcome these problems we recommend to use the `IEEEeqnarray`-environment.

## 4.2 IEEEeqnarray

The `IEEEeqnarray`-environment is a very powerful command with many options. Here, we will only introduce its basic functionalities. For more information we refer to Section 5 and the manual.<sup>3</sup>

First of all, in order to be able to use the `IEEEeqnarray`-environment one needs to include the package<sup>4</sup> `IEEEtrantools`. Include the following line in the header of your document:

```
\usepackage[retainorgcmds]{IEEEtrantools}
```

The strength of `IEEEeqnarray` is the possibility of specifying the number of *columns* in the equation array. Usually, this specification will be `{rCl}`, *i.e.*, three columns, the first column right-justified, the middle one centered with a little more space around it (therefore we specify capital C instead of lower-case c) and the third column left-justified:

<sup>3</sup>The official manual `IEEEtran.HOWTO.pdf` is distributed together with this short introduction. The part about `IEEEeqnarray` can be found in Appendix F.

<sup>4</sup>This package is also distributed together with this manual.



```
\begin{IEEEeqnarray}{rCl}
a & = & b + c \\
\\
& = & d + e + f + g + h \\
& + & i + j + k \nonumber \\
& & + l + m + n + o \\
\\
& = & p + q + r + s \\
\end{IEEEeqnarray}
```

$$a = b + c \quad (26)$$

$$= d + e + f + g + h + i + j + k \\ + l + m + n + o \quad (27)$$

$$= p + q + r + s \quad (28)$$

However, we can specify any number of needed columns. *E.g.*, `{c}` will give only one column with all entries centered, or `{rCl1}` will add a fourth, left-justified column, *e.g.*, for comments. Moreover, beside `l`, `c`, `r`, `L`, `C`, `R` for math mode entries there also exists `s`, `t`, `u` for left, centered, and right text mode entries, respectively. Moreover, we can add additional space by `.` and `/` and `?` in increasing order.<sup>5</sup>

Note that in contrast to `eqnarray` the spaces around the equality signs are correct!

### 4.3 Usage in Most Common Situations

In the following we will describe how we use `IEEEeqnarray` to solve the most common situations.

- If a line overlaps with the equation number as in (22), the command

```
\IEEEeqnarraynumspace
```

can be used: it has to be added in the corresponding line and makes sure that the whole equation array is shifted by the size of the equation numbers (the shift depends on the size of the number!): instead of

```
\begin{IEEEeqnarray}{rCl}
a & = & b + c \\
\\
& = & d + e + f + g + h \\
& + & i + j + k \\
\\
& = & l + m + n \\
\end{IEEEeqnarray}
```

$$a = b + c \quad (29)$$

$$= d + e + f + g + h + i + j + k \quad (30)$$

$$= l + m + n \quad (31)$$

we get

```
\begin{IEEEeqnarray}{rCl}
a & = & b + c \\
\\
& = & d + e + f + g + h \\
& + & i + j + k \\
\IEEEeqnarraynumspace \\
& = & l + m + n. \\
\end{IEEEeqnarray}
```

$$a = b + c \quad (32)$$

$$= d + e + f + g + h + i + j + k \quad (33)$$

$$= l + m + n. \quad (34)$$

<sup>5</sup>For more spacing types we refer to Sections 5.1 and 5.6 and the official manual.

- If the LHS is too long, as a replacement for the faulty `\lefteqn{}`-command, `IEEEeqnarray` offers the `\IEEEeqnarraymulticol`-command which works in all situations:

```
\begin{IEEEeqnarray}{rCl}
  \IEEEeqnarraymulticol{3}{l}{
    a + b + c + d + e + f
    + g + h
  }\nonumber\quad
  &= & i + j
\\
  &= & k + l + m
\end{IEEEeqnarray}
```

$$\begin{aligned}
 a + b + c + d + e + f + g + h \\
 &= i + j & (35) \\
 &= k + l + m & (36)
 \end{aligned}$$

The usage is identical to the `\multicolumns`-command in the `tabular`-environment. The first argument `{3}` specifies that three columns shall be combined to one which will be left-justified `{l}`.

Note that by adapting the `\quad`-command one can easily adapt the depth of the equation signs,<sup>6</sup> *e.g.*,

```
\begin{IEEEeqnarray}{rCl}
  \IEEEeqnarraymulticol{3}{l}{
    a + b + c + d + e + f
    + g + h
  }\nonumber\quad\quad\quad
  &= & i + j
\\
  &= & k + l + m
\end{IEEEeqnarray}
```

$$\begin{aligned}
 a + b + c + d + e + f + g + h \\
 &= i + j & (37) \\
 &= k + l + m & (38)
 \end{aligned}$$

- If an equation is split into two or more lines, L<sup>A</sup>T<sub>E</sub>X interprets the first `+` or `-` as sign instead of operator. Therefore, it is necessary to add an additional space `\:` between the operator and the term: instead of

```
\begin{IEEEeqnarray}{rCl}
  a &= & b + c
\\
  &= & d + e + f + g + h
  + i + j + k \nonumber\quad
  && + l + m + n + o
\\
  &= & p + q + r + s
\end{IEEEeqnarray}
```

$$\begin{aligned}
 a &= b + c & (39) \\
 &= d + e + f + g + h + i + j + k \\
 &\quad + l + m + n + o & (40) \\
 &= p + q + r + s & (41)
 \end{aligned}$$

we should write

---

<sup>6</sup>I think that one quad is the distance that looks good in most cases.

```

\begin{IEEEeqnarray}{rCl}
a & = & b + c \\
\\
& = & d + e + f + g + h \\
& + & i + j + k \nonumber \\
& & + l + m + n + o \\
\\
& = & p + q + r + s
\end{IEEEeqnarray}

```

$$a = b + c \quad (42)$$

$$= d + e + f + g + h + i + j + k + l + m + n + o \quad (43)$$

$$= p + q + r + s \quad (44)$$

(Compare the space between + and l!)

**Attention:** L<sup>A</sup>T<sub>E</sub>X is not completely silly: in certain situations like, *e.g.*, in front of

- an operator name like `\log`, `\sin`, `\det`, `\max`, *etc.*,
- an integral `\int` or sum `\sum`,
- a bracket with adaptive size using `\left` and `\right` (this is in contrast to normal brackets or brackets with fixed size like `\big(`),

a + or – cannot be a sign, but must be an operator. In those situations L<sup>A</sup>T<sub>E</sub>X will add the correct spacing and no additional space is needed.

▷ *Whenever you wrap a line, quickly check the result and verify that the spacing is correct!*

- If a particular line should not have an equation number, the number can be suppressed using `\nonumber` (or `\IEEEnonumber`). If on such a line a label `\label{eq:...}` is defined, then this label is passed on further to the next equation number that is not suppressed. However, it is recommended to put the labels right before the line-break `\\` or the end of the equation it belongs to. Apart from improving the readability of the source code this prevents a compilation error in the situation of a `\IEEEmulticol`-command after the label-definition.
- There also exists a \*-version where all equation numbers are suppressed. In this case an equation number can be made to appear using the command `\IEEEyesnumber`:

```

\begin{IEEEeqnarray*}{rCl}
a & = & b + c \\
& = & d + e \IEEEyesnumber \\
& = & f + g
\end{IEEEeqnarray*}

```

$$\begin{aligned}
 a &= b + c \\
 &= d + e \\
 &= f + g
 \end{aligned} \quad (45)$$

- Sub-numbers are also easily possible using the command `\IEEEyessubnumber`:

```

\begin{IEEEeqnarray}{rCl}
a & = & b + c \\
\IEEEyessubnumber \\
& = & d + e \\
\nonumber \\
& = & f + g \\
\IEEEyessubnumber \\
\end{IEEEeqnarray}

```

$$a = b + c \quad (46a)$$

$$= d + e$$

$$= f + g \quad (46b)$$

## 5 Advanced Typesetting

In this section we address a couple of more advanced typesetting problems and tools.

### 5.1 IEEEeqnarraybox: General Tables and Arrays

The package `IEEEtrantools` also provides the command `IEEEeqnarraybox`. This is basically the same as `IEEEeqnarray` but with the difference that it can be nested within other structures. Therefore it does not generate a full equation itself nor an equation number. It can be used both in text-mode (*e.g.*, inside a table) or in math-mode (*e.g.*, inside an equation). Hence, `IEEEeqnarraybox` is a replacement both for `array` and `tabular`. In case one does not want to let `IEEEeqnarraybox` to detect the mode automatically, but force one of these two modes, there are two subforms: `IEEEeqnarrayboxm` for math-mode and `IEEEeqnarrayboxt` for text-mode.

This is a silly table:

```

\begin{center}
\begin{IEEEeqnarraybox}{t.t.t}
\textbf{Item} & & \\
\textbf{Color} & & \\
\textbf{Number} & & \\
cars & green & 17 \\
trucks & red & 4 \\
bikes & blue & 25 \\
\end{IEEEeqnarraybox}
\end{center}

```

This is a silly table:

Item	Color	Number
cars	green	17
trucks	red	4
bikes	blue	25

Note that `t` in the argument of `IEEEeqnarraybox` stands for *centered text* and `.` adds space between the columns. Further possible arguments are `s` for *left text*, `u` for *right text*, `v` for a vertical line, and `V` for double vertical line. More details can be found in Tables IV and V on page 18 in the manual `IEEEtran_HOWTO.pdf`.

```
\begin{equation*}
P_U(u) = \left\{ \begin{array}{l}
\begin{IEEEeqnarraybox}[] [c] {1?s}
\IEEEstrut
0.1 & \text{if } u=0, \\
0.3 & \text{if } u=1, \\
0.6 & \text{if } u=2.
\end{IEEEstrut}
\end{IEEEeqnarraybox}
\right.
\end{equation*}
```

$$P_U(u) = \begin{cases} 0.1 & \text{if } u = 0, \\ 0.3 & \text{if } u = 1, \\ 0.6 & \text{if } u = 2. \end{cases}$$

Here ? is a large horizontal space between the columns, and `\IEEEstrut` adds a tiny little space above the first and below the bottom line. Moreover, note that the second optional argument `[c]` makes sure that the `IEEEeqnarraybox` is vertically centered. The other possible values for this option are `[t]` for aligning the first row with the surrounding baseline and `[b]` for aligning the bottom row with the surrounding baseline. Default is `[b]`, *i.e.*, if we do not specify this option, we get the following (in this case unwanted) result:

```
\begin{equation*}
P_U(u) = \left\{ \begin{array}{l}
\begin{IEEEeqnarraybox}{1?s}
0.1 & \text{if } u=0, \\
0.3 & \text{if } u=1, \\
0.6 & \text{if } u=2.
\end{IEEEeqnarraybox}
\right.
\end{equation*}
```

$$P_U(u) = \begin{cases} 0.1 & \text{if } u = 0, \\ 0.3 & \text{if } u = 1, \\ 0.6 & \text{if } u = 2. \end{cases}$$

We also dropped `\IEEEstrut` here with the result that the curly bracket is slightly too small at the top line. However, these manually placed `\IEEEstrut` commands are rather tiring. Moreover, when we would like to add vertical lines in a table, a first naive application of `IEEEeqnarraybox` yields the following:

```
\begin{equation*}
\begin{IEEEeqnarraybox}
{c'c;v;c'c'c}
D_1 & D_2 & & X_1 & X_2 & X_3 \\
\\ \hline
0 & 0 & & +1 & +1 & +1 \\
0 & 1 & & +1 & -1 & -1 \\
1 & 0 & & -1 & +1 & -1 \\
1 & 1 & & -1 & -1 & +1
\end{IEEEeqnarraybox}
\end{equation*}
```

$D_1$	$D_2$	$X_1$	$X_2$	$X_3$
0	0	+1	+1	+1
0	1	+1	-1	-1
1	0	-1	+1	-1
1	1	-1	-1	+1

We see that `IEEEeqnarraybox` makes a complete linebreak after each line. This is of course unwanted. Therefore, the command `\IEEEeqnarraystrutmode` is provided that switches the spacing system completely over to struts:

```

\begin{equation*}
\begin{IEEEeqnarraybox}[
  \IEEEeqnarraystrutmode
]{c'c;v;c'c'c}
D_1 & D_2 & & X_1 & X_2 & X_3 \\
\\ \hline
0 & 0 & & +1 & +1 & +1 \\
0 & 1 & & +1 & -1 & -1 \\
1 & 0 & & -1 & +1 & -1 \\
1 & 1 & & -1 & -1 & +1 \\
\end{IEEEeqnarraybox}
\end{equation*}

```

$D_1$	$D_2$	$X_1$	$X_2$	$X_3$
0	0	+1	+1	+1
0	1	+1	-1	-1
1	0	-1	+1	-1
1	1	-1	-1	+1

The strutmode also easily allows to ask for more “air” between each line:

```

\begin{equation*}
\begin{IEEEeqnarraybox}[
  \IEEEeqnarraystrutmode
  \IEEEeqnarraystrutsizadd{3pt}{1pt}
]{c'c/v/c'c'c}
D_1 & D_2 & & X_1 & X_2 & X_3 \\
\\ \hline
0 & 0 & & +1 & +1 & +1 \\
0 & 1 & & +1 & -1 & -1 \\
1 & 0 & & -1 & +1 & -1 \\
1 & 1 & & -1 & -1 & +1 \\
\end{IEEEeqnarraybox}
\end{equation*}

```

$D_1$	$D_2$	$X_1$	$X_2$	$X_3$
0	0	+1	+1	+1
0	1	+1	-1	-1
1	0	-1	+1	-1
1	1	-1	-1	+1

Here the first argument of `\IEEEeqnarraystrutsizadd{3pt}{1pt}` adds space above into each line, the second adds space below into each line.

## 5.2 Case-Distinctions

Case distinctions can be generated using `IEEEeqnarraybox` as shown in Section 5.1. However, in the standard situation the usage of `cases` is simpler and we therefore recommend to use this:

```

\begin{equation}
P_U(u) =
\begin{cases}
0.1 & \textnormal{if } u=0, \\
\\
0.3 & \textnormal{if } u=1, \\
\\
0.6 & \textnormal{if } u=2.
\end{cases}
\end{equation}

```

$$P_U(u) = \begin{cases} 0.1 & \text{if } u = 0, \\ 0.3 & \text{if } u = 1, \\ 0.6 & \text{if } u = 2. \end{cases} \quad (47)$$

For more complicated examples we do need to rely on `IEEEeqnarraybox`:

```

\begin{equation}
\left.
\begin{array}{l}
x = a + b \\
y = a - b
\end{array}
\right\} \Longleftrightarrow \begin{cases} a = \frac{x}{2} + \frac{y}{2} \\ b = \frac{x}{2} - \frac{y}{2} \end{cases} \quad (48)
\end{equation}

```

$$\left. \begin{array}{l} x = a + b \\ y = a - b \end{array} \right\} \Longleftrightarrow \begin{cases} a = \frac{x}{2} + \frac{y}{2} \\ b = \frac{x}{2} - \frac{y}{2} \end{cases} \quad (48)$$

### 5.3 Matrices

Matrices could be generated by `IEEEeqnarraybox`, however, the command `pmatrix` is easier to use:

```

\begin{equation}
\mat{P} =
\begin{pmatrix}
p_{11} & p_{12} & \ldots & p_{1n} \\
p_{21} & p_{22} & \ldots & p_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
p_{m1} & p_{m2} & \ldots & p_{mn}
\end{pmatrix}
\end{equation}

```

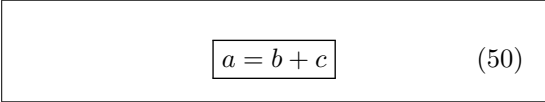
$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{pmatrix} \quad (49)$$

Note that it is not necessary to specify the number of columns (or rows) in advance. More possibilities are `bmatrix` (for matrices with square brackets), `Bmatrix` (curly brackets), `vmatrix` ( $|$ ), `Vmatrix` ( $\|$ ), and `matrix` (no brackets at all).

## 5.4 Framed Equations

To generate equations that are framed one can use the `\boxed{...}`-command. However, usually this will yield a too tight frame around the equation:

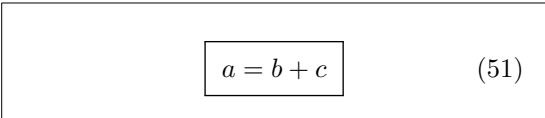
```
\begin{equation}
  \boxed{
    a = b + c
  }
\end{equation}
```



$$a = b + c \quad (50)$$

To give the frame a little bit more “air” we need to redefine the length-variable `\fboxsep`. We do this in a way that restores its original definition afterwards:

```
\begin{equation}
  \newlength{\fboxstore}
  \setlength{\fboxstore}{\fboxsep}
  \setlength{\fboxsep}{6pt}
  \boxed{
    a = b + c
  }
  \setlength{\fboxsep}{\fboxstore}
\end{equation}
```



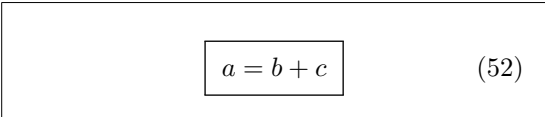
$$a = b + c \quad (51)$$

Note that the `\newlength`-command must be given only once per document. To ease one’s life, we recommend to define a macro for this in the document header:

```
\newlength{\eqboxstorage}
\newcommand{\eqbox}[1]{
  \setlength{\eqboxstorage}{\fboxsep}
  \setlength{\fboxsep}{6pt}
  \boxed{#1}
  \setlength{\fboxsep}{\eqboxstorage}
}
```

Now the framed equation can be produced as follows:

```
\begin{equation}
  \eqbox{
    a = b + c
  }
\end{equation}
```



$$a = b + c \quad (52)$$

In case of `multline` or `IEEEeqnarray` this approach does not work because the `\boxed{...}` command does not allow line breaks or similar. Therefore we need to rely on `IEEEeqnarraybox` for boxes around equations on several lines:



```

\begin{equation}
\eqbox{
\begin{IEEEeqnarraybox}{rCl}
a & = & b + c
\\
& = & d + e + f + g + h
+ i + j + k \\
& + & l + m + n + o
\\
& = & p + q + r + s
\end{IEEEeqnarraybox}
}
\end{equation}

```

$$\begin{array}{rcl}
 a & = & b + c \\
 & = & d + e + f + g + h + i + j + k \\
 & & + l + m + n + o \\
 & = & p + q + r + s
 \end{array} \tag{53}$$

Some comments:

- The basic idea here is to replace the original `IEEEeqnarray` command by a `IEEEeqnarraybox` and then wrap everything into an `equation`-environment.
- The equation number is produced by the surrounding `equation`-environment. If we would like to have the equation number vertically centered, we need to center the `IEEEeqnarraybox`:

```

\begin{equation}
\eqbox{
\begin{IEEEeqnarraybox}[] [c]{rCl}
a & = & b + c + d + e
+ f + g + h
\\
& + & i + j + k + l
+ m + n
\\
& + & o + p + q
\end{IEEEeqnarraybox}
}
\end{equation}

```

$$\begin{array}{rcl}
 a & = & b + c + d + e + f + g + h \\
 & & + i + j + k + l + m + n \\
 & & + o + p + q
 \end{array} \tag{54}$$

in contrast to

```

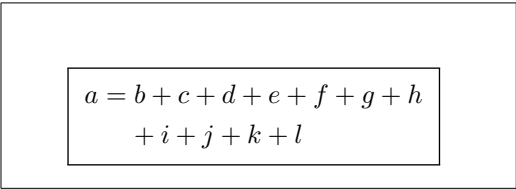
\begin{equation}
\eqbox{
\begin{IEEEeqnarraybox}{rCl}
a & = & b + c + d + e
+ f + g + h
\\
& + & i + j + k + l
+ m + n
\\
& + & o + p + q
\end{IEEEeqnarraybox}
}
\end{equation}

```

$$\begin{array}{rcl}
 a & = & b + c + d + e + f + g + h \\
 & & + i + j + k + l + m + n \\
 & & + o + p + q
 \end{array} \tag{55}$$

- When changing the `IEEEeqnarray` into a `IEEEeqnarraybox`, be careful to delete any remaining `\nonumber` commands inside of the `IEEEeqnarraybox`! Since `IEEEeqnarraybox` does not know equation numbers anyway, any remaining `\nonumber` command will “leak” through and prevent `equation` to put a number!

```
\begin{equation}
\eqbox{
\begin{IEEEeqnarraybox}{rCl}
a & = & b + c + d + e
+ f + g + h \nonumber \\
& & i + j + k + l
\end{IEEEeqnarraybox}
}
\end{equation}
```



$$a = b + c + d + e + f + g + h + i + j + k + l$$

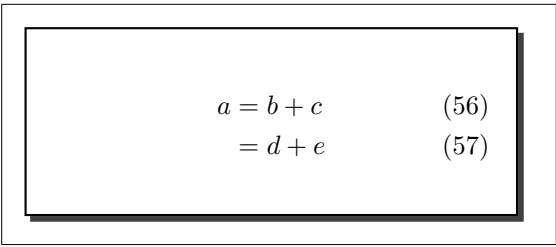
## 5.5 More Fancy Frames

More fancy frames can be produced using the `framed` and the `pstricks` packages.<sup>7</sup> Use the following commands in the header of your document:

```
\usepackage{pstricks,framed}
```

Then we can produce all kinds of fancy frames:

```
\renewcommand{\FrameCommand}{%
\psshadowbox[shadowsize=0.3em,%
framesep=1.0em, fillstyle=solid,%
fillcolor=white]}
\begin{framed}
\begin{IEEEeqnarray}{rCl}
a & = & b + c
\\
& & = d + e
\end{IEEEeqnarray}
\end{framed}
```



$$a = b + c \quad (56)$$

$$= d + e \quad (57)$$

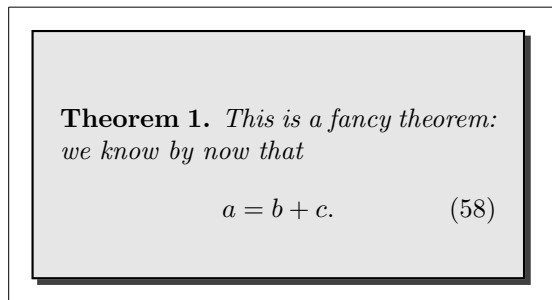
The frame can also be put around larger structures like theorems:

<sup>7</sup>This will not work with pdf<sub>l</sub>at<sub>e</sub>x. To produce PDF one needs to go via dvips and ps2pdf.

```

\newgray{mygray}{0.9}
\renewcommand{\FrameCommand}{%
\psshadowbox[shadowsize=0.3em,%
framesep=1.0em, fillstyle=solid,%
fillcolor=mygray]}
\begin{framed}
  \begin{theorem}
    This is a fancy theorem:
    we know by now that
    \begin{equation}
      a = b + c.
    \end{equation}
  \end{theorem}
\end{framed}

```



Note that in this example we have assumed that the `theorem`-environment has been defined in the header:

```

\usepackage{amsthm}
\newtheorem{theorem}{Theorem}

```

## 5.6 Putting the QED Right

The package `amsthm` that we have used in Section 5.5 to generate a theorem actually also defines a `proof`-environment:

```

\begin{proof}
  This is the proof of some
  theorem. Once the proof is
  finished we put a white box
  at the end to denote QED.
\end{proof}

```

*Proof.* This is the proof of some theorem.  
Once the proof is finished we put a white  
box at the end to denote QED. □

The QED-symbol should be put on the last line of the proof. However, if the last line is an equation, then this is done wrongly:

```

\begin{proof}
  This is a proof that ends
  with an equation:
  \begin{equation*}
    a = b + c.
  \end{equation*}
\end{proof}

```

*Proof.* This is a proof that ends with an  
equation:

$$a = b + c.$$

□

In such a case, the QED-symbol must be put by hand using the command `\qedhere`:

```

\begin{proof}
  This is a proof that ends
  with an equation:
  \begin{equation*}
    a = b + c. \qedhere
  \end{equation*}
\end{proof}

```

*Proof.* This is a proof that ends with an  
equation:

$$a = b + c.$$

□

Unfortunately, this correction does not work for `IEEEeqnarray`:

```
\begin{proof}
  This is a proof that ends
  with an equation array:
  \begin{IEEEeqnarray*}{rCl}
    a & = & b + c \\
    & = & d + e. \qedhere
  \end{IEEEeqnarray*}
\end{proof}
```

*Proof.* This is a proof that ends with an equation array:

$$\begin{aligned} a &= b + c \\ &= d + e. \quad \square \end{aligned}$$

The reason for this is the internal structure of `IEEEeqnarray`: it always puts two invisible columns at both sides of the array that only contain a stretchable space. By this `IEEEeqnarray` ensures that the equation array is horizontally centered. The `\qedhere`-command should actually be put *outside* this stretchable space, but this does not happen as these columns are invisible to the user.

There is, however, a very simple remedy: we define these stretching columns ourselves!

```
\begin{proof}
  This is a proof that ends
  with an equation array:
  \begin{IEEEeqnarray*}{+rCl+x*}
    a & = & b + c \\
    & = & d + e. & \qedhere
  \end{IEEEeqnarray*}
\end{proof}
```

*Proof.* This is a proof that ends with an equation array:

$$\begin{aligned} a &= b + c \\ &= d + e. \quad \square \end{aligned}$$

Here the `+` in `{+rCl+x*}` denotes stretchable spaces, one on the left of the equations (which, if not specified, will be done automatically by `IEEEeqnarray`!) and one on the right of the equations. But now on the right, *after* the stretching column, we add an empty column `x`. This column will be only needed on the last line when we will put the `\qedhere`-command there. Finally, we specify a `*`. This is a null-space that prevents `IEEEeqnarray` to add another unwanted `+`-space!

In case of equation numbering, we have a similar problem. If you compare

```
\begin{proof}
  This is a proof that ends
  with a numbered equation:
  \begin{equation}
    a = b + c.
  \end{equation}
\end{proof}
```

*Proof.* This is a proof that ends with a numbered equation:

$$a = b + c. \quad (59)$$

□

with

```
\begin{proof}
  This is a proof that ends
  with a numbered equation:
  \begin{equation}
    a = b + c. \qedhere
  \end{equation}
\end{proof}
```

*Proof.* This is a proof that ends with a numbered equation:

$$a = b + c. \quad (60)$$

□

you notice that in the (correct) second version the □ is much closer to the equation than in the first version.

Similarly, the correct way of putting the QED-symbol at the end of an equation array is as follows:

```
\begin{proof}
  This is a proof that ends
  with an equation array:
  \begin{IEEEeqnarray}{+rCl+x*}
    a & = & b + c \\
    & = & d + e. \\
    & & \qedhere\nonumber
  \end{IEEEeqnarray}
\end{proof}
```

*Proof.* This is a proof that ends with an equation array:

$$a = b + c \quad (61)$$

$$= d + e. \quad (62)$$

□

which contrasts with

```
\begin{proof}
  This is a proof that ends
  with an equation array:
  \begin{IEEEeqnarray}{rCl}
    a & = & b + c \\
    & = & d + e.
  \end{IEEEeqnarray}
\end{proof}
```

*Proof.* This is a proof that ends with an equation array:

$$a = b + c \quad (63)$$

$$= d + e. \quad (64)$$

□

A small comment at the end: The IEEEtran-class does not allow for the command `\qedhere`. The reason is that the IEEE Transactions do not want to have the QED-symbol put onto the same line. One can, however, force this behavior. To do this, one needs to use `\IEEEQED` instead of `\qedhere` and add just before the `\end{IEEEproof}` the command `\let\IEEEQED\relax` that will suppress the usual output of the QED-symbol for a second time. Example:

```
\begin{IEEEproof}
  For the IEEE Transactions one should use the IEEEproof environment:
  \begin{IEEEeqnarray}{rCl+x*}
    a & = & b + c \\
    & & \IEEEQED\nonumber
  \end{IEEEeqnarray}
  \let\IEEEQED\relax %This suppresses the output of the QED-symbol.
\end{IEEEproof}
```

## 6 Emacs and IEEEeqnarray

When working with Emacs you can ease your life by defining a few new commands. In the `dot_emacs`-file that comes together with this document the following commands are defined:

- **Control-c i**: Insert a standard `IEEEeqnarray`-environment (similar to `control-c control-e`).
- **Control-c o**: As `control-c i`, but the `*`-version.
- **Control-c b**: Add a line break at a specific place. This is very helpful in editing too long lines. Suppose you have typed the following L<sup>A</sup>T<sub>E</sub>X-code:

```
\begin{IEEEeqnarray}{rCl}
  a & = & b + c \\
  & = & d + e + f + g + h + i
    + j + k + l + m + n + o
\end{IEEEeqnarray}
```

$$\begin{aligned} a &= b + c \\ &= d + e + f + g + h + i + j + k + l + m + n \end{aligned} \quad (65)$$

After compiling you realize that you have to break the line before  $l$ . You now just have to put the cursor on the  $+$ -sign in front of  $l$  and press `control-c b`. Then the line is wrapped there and also the additional space `\:` is added at the right place:

```
\begin{IEEEeqnarray}{rCl}
  a & = & b + c \\
  & = & d + e + f + g + h + i
    + j + k \nonumber\\
  & & + l + m + n + o
\end{IEEEeqnarray}
```

$$\begin{aligned} a &= b + c \\ &= d + e + f + g + h + i + j + k \\ &\quad + l + m + n + o \end{aligned} \quad \begin{matrix} (67) \\ (68) \end{matrix}$$

- **Control-c n**: As `Control-c b`, but without adding the additional space `\:`.
- **Control-c m**: Insert a `IEEEeqnarraymulticol`-command. This is very helpful when the LHS is too long. Suppose you have typed the following L<sup>A</sup>T<sub>E</sub>X-code:

```
\begin{IEEEeqnarray}{rCl}
  a + b + c + d + e + f
  + g + h + i + j
  & = & k + l \\
  & = & m + n
\end{IEEEeqnarray}
```

$$\begin{aligned} a + b + c + d + e + f + g + h + i + j &= k \\ &= m \end{aligned} \quad \begin{matrix} (69) \\ (70) \end{matrix}$$

After compiling you realize that the LHS is too long. You now just have to put the cursor somewhere on the first line and type `control-c m`. Then you get

```

\begin{IEEEeqnarray}{rCl}
\IEEEeqnarraymulticol{3}{1}{
  a + b + c + d + e + f
  + g + h + i + j
}\nonumber \\ \quad
& = & k + l \\
& = & m + n
\end{IEEEeqnarray}

```

$$a + b + c + d + e + f + g + h + i + j$$

$$= k + l \quad (71)$$

$$= m + n \quad (72)$$

- Furthermore, in the `dot_emacs`-file definitions are given for the `ispell`-command to ignore the `IEEEeqnarray`-environment. This simplifies the spell-check considerably (otherwise, *e.g.*, `{rCl}` is regarded as miss-spelled expression).

## 7 Some Final Remarks

The “rules” stated in this document are purely based on my own experience with typesetting L<sup>A</sup>T<sub>E</sub>X in my publication papers and my Ph.D. thesis, and on my—some people might say unfortunate—habit of always incorporating a lot of mathematical expressions in there...

If you encounter any situation that seems to contradict the suggestions of this document, then I would be very happy if you could send me a corresponding L<sup>A</sup>T<sub>E</sub>X- or PDF-file! As a matter of fact, any kind of feedback, criticism, suggestion, *etc.*, is very much welcome! Write to:

`stefan dot moser at ieee dot org`

Thanks!

Stefan M. Moser