How to Typeset Equations in LaTeX

Stefan M. Moser

3 January 2013
Version 4.0

Contents

1 Introduction

LaTeX is a very powerful tool for typesetting in general and for typesetting math in particular. In spite of its power, however, 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 LaTeX.

Note that this document is based on the newest version of `IEEEtran`-package (`IEEEtrantools.sty` version 1.3 or `IEEEtran.cls` version 1.8).

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 Sections 4 and 5: there we introduce the powerful `IEEEeqnarray`-environment that should be used in any case instead of `align` or `eqnarray`.

In Section 6 some more advanced problems and possible solutions are discussed, and Section 7 contains some hints and tricks about the editor Emacs.

In the following any LaTeX 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 our 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`: LaTeX source file of this manual.
- `dot_emacs`: commands to be included in the preference file of Emacs (`.emacs`) (see Section 7).
- `IEEEtrantools.sty` [2012/12/27 V1.3 by Michael Shell]: package needed for the `IEEEeqnarray`-environment.
- `IEEEtran.cls` [2012/12/27 V1.8 by Michael Shell]: LaTeX document class package for papers in IEEE format.
- `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 LaTeX concerning typesetting of mathematics is based on the package `amsmath`. Every current distribution of LaTeX 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}
```

$$a = b + c \tag{1}$$

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: 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[1]) 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.

[1] A possible exception is if one decides to use `IEEEeqnarray` for *all* mathematical expressions; see the discussion in Section 4.3.

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:[2]

```
\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.

However, in spite of its ease of 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 the RHS 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 over the plus operator in front of k. The better solution is provided by the `IEEEeqnarray`-environment that will be discussed in detail in Sections 4 and 5:

[2] As a reminder: it is necessary to include the `amsmath`-package for this command to work!

```
\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}
```

$$\begin{aligned} a &= b + c + d + e + f + g + h + i + j \\ &\quad + k + l + m + n + o \end{aligned} \tag{5}$$

In this case the second line is horizontally 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.

Also note that `multline` wrongly forces a minimum spacing on the left of the first line even if it has not enough space on the right, causing a noncentered equation. This can even lead to the very ugly typesetting where the second line containing the RHS of an equality is actually *to the left* of the first line containing the LHS:

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

$$\begin{aligned} &a + b + c + d + e + f + g + h + i + j \\ &= k + l + m + n + o + p + q + r + s + t + u \end{aligned} \tag{6}$$

For this reason we give the following rule:

> The `multline`-environment should exclusively be used in the five specific situations described in Sections 3.1–3.5.

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}
```

$$\begin{aligned} \lim_{n\uparrow\infty} \Big\{ & a + b + c + d + e + f \\ & + g + h + i + j + k + l \\ & + m + n + o + p + q \Big\} \end{aligned} \tag{7}$$

3.2 Case 2: A single equation to be wrapped before an 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}
```

$$a+b+c+d+e+f = g+h+i+j+k+l+m \quad (8)$$

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{for } 0 \le n
  \le n_{\textnormal{max}}
\end{multline}
```

$$a+b+c+d = e+f+g+h, \quad \text{for } 0 \le n \le n_{\max} \quad (9)$$

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 \\+ h + i + j
  + k + l = m
\end{multline}
```

$$a+b+c+d+e+f+g \\ +h+i+j+k+l = m \quad (10)$$

3.5 Case 5: A term on the 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^{-}(X|Y) \le \frac{n+1}{e}
  - h(X|Y)
  \\
  + \int p(y) \log \left(
    \frac{\mathsf{E}\big[|X|^2
      \big| Y=y\big]}{n}
  \right) \textnormal{d} y
\end{multline}
```

$$h^-(X|Y) \le \frac{n+1}{e} - h(X|Y) \\ + \int p(y) \log \left(\frac{\mathsf{E}\big[|X|^2 \big| Y=y\big]}{n} \right) \mathrm{d}y \quad (11)$$

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}{l}{
    h^{-}(X|Y)
  }\nonumber\\;\;
  & \le & \frac{n+1}{e}
  - h(X|Y) \nonumber\\
  && + \int p(y) \log \left(
    \frac{\mathsf{E}\big[|X|^2
      \big| Y=y\big]}{n}
  \right) \textnormal{d} y
  \nonumber\\
\end{IEEEeqnarray}
```

$$\begin{aligned} &h^-(X|Y)\\ &\quad\le \frac{n+1}{e} - h(X|Y)\\ &\qquad + \int p(y) \log \left(\frac{\mathsf{E}\big[|X|^2\big|Y=y\big]}{n} \right) \mathrm{d}y \end{aligned} \tag{12}$$

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 horizontal 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[3] could be used:

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

$$a = b + c \tag{13}$$
$$= d + e \tag{14}$$

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}
```

$$a = b + c \tag{15}$$
$$= d + e + f + g + h + i + j + k + l$$
$$+ m + n + o \tag{16}$$
$$= p + q + r + s \tag{17}$$

[3] 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`}.

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}
```

$$\begin{aligned} a &= b+c && (18)\\ &= d+e+f+g+h+i+j+k+l \\ &\quad +m+n+o && (19)\\ &= p+q+r+s && (20) \end{aligned}$$

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}
```

$$a = a = a \qquad (21)$$

- 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}
```

$$\begin{aligned} a &= b+c && (22)\\ &= d+e+f+g+h^2+i^2+j && (23) \end{aligned}$$

- 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}
```

$$\begin{aligned} &a+b+c+d+e+f+g+h \\ &\quad = i+j+k+l+m && (24)\\ &\quad = n+o+p+q+r+s && (25) \end{aligned}$$

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}
```

$$a+b+c+d+e+f+g+h$$
$$= \quad i+j \tag{26}$$

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

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

4.2 Solution: basic usage of IEEEeqnarray

The `IEEEeqnarray`-environment is a very powerful command with many options. In this manual we will only introduce some of the most important functionalities. For more information we refer to the official manual.[4] First of all, in order to be able to use the `IEEEeqnarray`-environment, one needs to include the package[5] `IEEEtrantools`. Include the following line in the header of your document:

```
\usepackage{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:

```
\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}
```

$$a = b+c \tag{27}$$
$$= d+e+f+g+h+i+j+k$$
$$\quad +l+m+n+o \tag{28}$$
$$= p+q+r+s \tag{29}$$

However, we can specify any number of needed columns. For example, `{c}` will give only one column (which is centered) or `{rCll}` will add a fourth, left-justified column, e.g., for additional specifications. 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 `?` and `"` in

[4] The official manual `IEEEtran_HOWTO.pdf` is distributed together with this short introduction. The part about `IEEEeqnarray` can be found in Appendix F.

[5] This package is also distributed together with this manual. Note that if the document uses the `IEEEtran`-class, then `IEEEtrantools` is loaded automatically and must not be included.

increasing order.[6] More details about the usage of `IEEEeqnarray` will be given in Section 5.

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

4.3 A remark about consistency

There are two more issues that have not been mentioned so far, but that might cause some inconsistencies when all three environments, `equation`, `multline`, and `IEEEeqnarray`, are used intermixedly:

- `multline` allows for an equation starting on top of a page, while `equation` and `IEEEeqnarray` try to put a line of text first, before the equation starts.
- `equation` sometimes forces the equation number onto the next line, even if there was still enough space available on the line.

If one is unhappy with this, one can actually resort to using `IEEEeqnarray` exclusively in all situations. To mimic the normal `equation`-environment we then use `IEEEeqnarray` with only one column `{c}`. Emulating `multline` is slightly more complicated: we implement `IEEEeqnarray` with only one column `{l}`, and use after the line-break(s) either `\IEEEeqnarraymulticol` to adapt the column type of the new line or manually add some shift to the right.

5 More Details about IEEEeqnarray

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

5.1 Shift to the left: IEEEeqnarraynumspace

If a line overlaps with the equation number as in (23), 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 + o
\end{IEEEeqnarray}
```

$$\begin{aligned} a &= b + c && (30)\\ &= d + e + f + g + h + i + j + k + l && (31)\\ &= m + n + o && (32) \end{aligned}$$

[6] For examples of their usage we refer to Section 6.1. More spacing types can be found in the examples given in Sections 5.3 and 6.8, and in the official manual.

we get

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

$$\begin{aligned} a &= b + c && (33)\\ &= d + e + f + g + h + i + j + k + l && (34)\\ &= m + n + o && (35) \end{aligned}$$

5.2 First line too long: IEEEeqnarraymulticol

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}
```

$$\begin{aligned} &a + b + c + d + e + f + g + h && \\ &\quad = i + j && (36)\\ &\quad = k + l + m && (37) \end{aligned}$$

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

Note that by adapting the `\quad`-command one can easily adapt the depth of the equation signs,[7] e.g.,

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

$$\begin{aligned} &a + b + c + d + e + f + g + h && \\ &\qquad\qquad = i + j && (38)\\ &\qquad\qquad = k + l + m && (39) \end{aligned}$$

Note that `\IEEEeqnarraymulticol` must be the first command in a cell. This is usually no problem; however, it might be the cause of some strange compilation errors. For example, one might put a `\label`-command on the first line inside[8] of `IEEEeqnarray`, which is OK in general, but not OK if it is followed by the `\IEEEeqnarraymulticol`-command.

[7] I think that one quad is the distance that looks good in most cases.

[8] I strongly recommend to put all labels at the end of the corresponding equation; see Section 5.4.

5.3 Line-break: unary versus binary operators

If an equation is split into two or more lines, LaTeX interprets the first + or − as a sign instead of an 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\\
  && + l + m + n + o
  \\
  & = & p + q + r + s
\end{IEEEeqnarray}
```

$$\begin{aligned} a &= b + c && (40)\\ &= d + e + f + g + h + i + j + k \\ &\quad + l + m + n + o && (41)\\ &= p + q + r + s && (42) \end{aligned}$$

we should write

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

$$\begin{aligned} a &= b + c && (43)\\ &= d + e + f + g + h + i + j + k \\ &\quad + l + m + n + o && (44)\\ &= p + q + r + s && (45) \end{aligned}$$

(Compare the space between + and l!)

Attention: The distinction between the *unary operator* (sign) and the *binary operator* (addition/subtraction) is not satisfactorily solved in LaTeX.[9] In some cases LaTeX will automatically assume that the operator cannot be unary and will therefore add additional spacing. This happens, 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(`).

This decision, however, might be faulty. E.g., it makes perfect sense to have a unary operator in front of the logarithm:

```
\begin{IEEEeqnarray*}{rCl"s}
  \log \frac{1}{a}
  & = & -\log a
  & (binary, wrong) \\
  & = & -{\log a}
  & (unary, correct)
\end{IEEEeqnarray*}
```

$$\begin{aligned} \log \frac{1}{a} &= -\log a && \text{(binary, wrong)}\\ &= -\log a && \text{(unary, correct)} \end{aligned}$$

[9] The problem actually goes back to TeX.

In this case you have to correct it manually. Unfortunately, there is no clean way of doing this. To enforce a unary operator, enclosing the expression following the unary operator and/or the unary operator itself into curly brackets `{...}` will usually work. For the opposite direction, i.e., to enforce a binary operator (as, e.g., needed in (44)), the only option is to put in the correct space (`\:`) manually.

Compare in the following example the spacing between the first minus-sign on the RHS and b (or $\log b$):

```
\begin{IEEEeqnarray*}{rCl's}
  a & = & - b - b - c
  & (default unary) \\
  & = & {-} {b} - b - c
  & (default unary, no effect) \\
  & = & -\: b - b - c
  & (changed to binary) \\
  & = & - \log b - b - d
  & (default binary) \\
  & = & {-} {\log b} - b - d
  & (changed to unary) \\
  & = & - \log b - b {-} d
  & (changed $-d$ to unary)
\end{IEEEeqnarray*}
```

$$
\begin{aligned}
a &= -b - b - c && \text{(default unary)}\\
&= -b - b - c && \text{(default unary, no effect)}\\
&= - b - b - c && \text{(changed to binary)}\\
&= - \log b - b - d && \text{(default binary)}\\
&= -\log b - b - d && \text{(changed to unary)}\\
&= - \log b - b{-}d && \text{(changed } -d \text{ to unary)}
\end{aligned}
$$

We learn:

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

5.4 Equation-numbering

While `IEEEeqnarray` assigns an equation number to all lines, the starred version `IEEEeqnarray*` suppresses all numbers. This behavior can be changed individually per line by the commands

`\IEEEyesnumber` and `\IEEEnonumber` (or `\nonumber`).

For subnumbering the corresponding commands

`\IEEEyessubnumber` and `\IEEEnosubnumber`

are available. These four commands only affect the line on which they are invoked, however, there also exist starred versions

`\IEEEyesnumber*`, `\IEEEnonumber*`,
`\IEEEyessubnumber*`, `\IEEEnosubnumber*`

that will remain active over several lines until another starred command is invoked.

Consider the following example.

```
\begin{IEEEeqnarray*}{rCl}
  a & = & b_{1}    \\
  & = & b_{2}    \IEEEyesnumber\\
  & = & b_{3}      \\
  & = & b_{4}    \IEEEyesnumber*\\
  & = & b_{5}      \\
  & = & b_{6}      \\
  & = & b_{7}    \IEEEnonumber\\
  & = & b_{8}      \\
  & = & b_{9}    \IEEEnonumber*\\
  & = & b_{10}     \\
  & = & b_{11}   \IEEEyessubnumber*\\
  & = & b_{12}     \\
  & = & b_{13}   \IEEEyesnumber\\
  & = & b_{14}     \\
  & = & b_{15}
\end{IEEEeqnarray*}
\begin{IEEEeqnarray}{rCl}
  \label{eq:bad_placement}
  a & = & b_{16}\IEEEyessubnumber*\\
  & = & b_{17}     \\
  & = & b_{18}   \IEEEyesnumber
                  \IEEEyessubnumber*\\
  & = & b_{19}     \\
  & = & b_{20}   \IEEEnosubnumber*\\
  & = & b_{21}     \\
  & = & b_{22}   \nonumber\\
  & = & b_{23}
\end{IEEEeqnarray}
\begin{IEEEeqnarray}{rCl}
  \IEEEyesnumber\IEEEyessubnumber*
  a & = & b_{24} \\
  & = & b_{25}
  \label{eq:good_placement}\\
  & = & b_{26}
\end{IEEEeqnarray}
```

$$
\begin{aligned}
a &= b_{1} \\
&= b_{2} && (46)\\
&= b_{3} \\
&= b_{4} && (47)\\
&= b_{5} && (48)\\
&= b_{6} && (49)\\
&= b_{7} \\
&= b_{8} && (50)\\
&= b_{9} \\
&= b_{10} \\
&= b_{11} && (50a)\\
&= b_{12} && (50b)\\
&= b_{13} && (51)\\
&= b_{14} && (51a)\\
&= b_{15} && (51b)
\end{aligned}
$$

$$
\begin{aligned}
a &= b_{16} && (51c)\\
&= b_{17} && (51d)\\
&= b_{18} && (52a)\\
&= b_{19} && (52b)\\
&= b_{20} && (53)\\
&= b_{21} && (54)\\
&= b_{22} \\
&= b_{23} && (55)
\end{aligned}
$$

$$
\begin{aligned}
a &= b_{24} && (56a)\\
&= b_{25} && (56b)\\
&= b_{26} && (56c)
\end{aligned}
$$

Note that the behavior in the line 13 (i.e., the line containing b_{13}) is probably unwanted: there the command `\IEEEyesnumber` temporarily switches to a normal equation number (implicitly resetting the subnumbers), but in the subsequent line the `\IEEEyessubnumber*` from line 11 takes control again, i.e., subnumbering is reactivated. The correct way of increasing the number and start directly with a new subnumber is shown in line 18 and in line 24. Also note that the subnumbering even works across different `IEEEeqnarray` as can be seen in line 16.

The best way of understanding the numbering behavior is to note that in spite of the eight different commands, there are only three different modes:

1. No equation number (corresponding to `\IEEEnonumber`).
2. A normal equation number (corresponding to `\IEEEyesnumber`): the equation

counter is incremented and then displayed.

3. An equation number with subnumber (corresponding to `\IEEEyessubnumber`): only the subequation counter is incremented and then both the equation and the subequation numbers are displayed. (*Attention:* If the equation number shall be incremented as well, which is usually the case for the start of a new subnumbering, then also `\IEEEyesnumber` has to be given!)

The understanding of the working of these three modes is also important when using labels to refer to equations. Note that the label must always be given *after* the `\IEEEyesnumber` or `\IEEEyessubnumber` command as only then the counters have been increased to the correct value. Otherwise, a label can produce an undesired output: for example a reference to label `eq:bad_placement` in line 16 will wrongly result[10] in (52). We learn:

> Labels should always be put as last command right in front of the line-break \\ or the end of the equation it belongs to.

Besides preventing unwanted results, this rules also increases the readability of the source code and prevents a compilation error in the situation of a `\IEEEeqnarraymulticol`-command after a label-definition. A correct example is shown in (56b).

5.5 Page-breaks inside of IEEEeqnarray

By default, `amsmath` does not allow page-breaks within multiple equations. This is usually too restrictive, particularly, if a document contains long equation arrays. This behavior can be changed by putting the following line into the document header:

```
\interdisplaylinepenalty=xx
```

Here, `xx` is some number: the larger this number, the less likely it is that an equation array is broken over to the next page. So, a value 0 fully allows page-breaks, a value 2500 allows page-breaks, but only if LaTeX finds no better solution, or a value 9999 basically prevents page-breaks.[11]

6 Advanced Typesetting

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

[10] To understand its value note that when it was invoked, subnumbering was deactivated. So the label only refers to a normal equation number. However, no such number was active there either, so the label is passed on to line 19 where the equation counter is incremented for the first time.

[11] I usually use a value 1000 that in principle allows page-breaks, but still asks LaTeX to check if there is no other way.

6.1 IEEEeqnarraybox: general tables and arrays

The package `IEEEtrantools` also provides the environment `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`.[12]

```
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 a vertical double-line. More details can be found in Tables IV and V on page 18 in the manual `IEEEtran_HOWTO.pdf`.

Another example:[13]

```
\begin{equation}
  P_U(u) = \left\{ \,
    \begin{IEEEeqnarraybox}[][c]{l?s}
      \IEEEstrut
      0.1 & if $u=0$, \\
      0.3 & if $u=1$, \\
      0.6 & if $u=2$.
      \IEEEstrut
    \end{IEEEeqnarraybox}
  \right.
  \label{eq:example_left_right1}
\end{equation}
```

$$P_U(u) = \begin{cases} 0.1 & \text{if } u = 0, \\ 0.3 & \text{if } u = 1, \\ 0.6 & \text{if } u = 2. \end{cases} \tag{57}$$

Here `?` is a large horizontal space between the columns, and `\IEEEstrut` adds a tiny 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

[12] In case one does not want to let `IEEEeqnarraybox` to detect the mode automatically, but to force one of these two modes, there are two subforms: `IEEEeqnarrayboxm` for math-mode and `IEEEeqnarrayboxt` for text-mode.

[13] For another way of generating case distinctions, see Section 6.2.

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{IEEEeqnarraybox}{l?s}
      0.1 & if $u=0$, \\
      0.3 & if $u=1$, \\
      0.6 & 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.

Actually, 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 line-break 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 and thereby eliminating the need of manually adding `\IEEEstrut`:

```
\begin{equation*}
  \begin{IEEEeqnarraybox}[
     \IEEEeqnarraystrutmode
     \IEEEeqnarraystrutsizeadd{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 `\IEEEeqnarraystrutsizeadd{3pt}{1pt}` adds space above into each line, the second adds space below into each line.

6.2 Case distinctions

Case distinctions can be generated using `IEEEeqnarraybox` as shown in Section 6.1. 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 & \text{if } u=0,
    \\
    0.3 & \text{if } u=1,
    \\
    0.6 & \text{if } u=2.
  \end{cases}
\end{equation}
```

$$P_U(u) = \begin{cases} 0.1 & \text{if } u=0, \\ 0.3 & \text{if } u=1, \\ 0.6 & \text{if } u=2. \end{cases} \tag{58}$$

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

```
\begin{equation}
 \left.
  \begin{IEEEeqnarraybox}[
     \IEEEeqnarraystrutmode
     \IEEEeqnarraystrutsizeadd{2pt}
      {2pt}
    ][c]{rCl}
   x & = & a + b\\
   y & = & a - b
  \end{IEEEeqnarraybox}
  \, \right\} \quad
 \Longleftrightarrow \quad
 \left\{ \,
  \begin{IEEEeqnarraybox}[
     \IEEEeqnarraystrutmode
     \IEEEeqnarraystrutsizeadd{7pt}
      {7pt}
    ][c]{rCl}
   a & = & \frac{x}{2}
   + \frac{y}{2}
   \\
   b & = & \frac{x}{2}
   - \frac{y}{2}
  \end{IEEEeqnarraybox}
 \right.
 \label{eq:example_left_right2}
\end{equation}
```

$$\left.\begin{aligned} x &= a+b \\ y &= a-b \end{aligned}\right\} \quad \Longleftrightarrow \quad \left\{\begin{aligned} a &= \frac{x}{2}+\frac{y}{2} \\ b &= \frac{x}{2}-\frac{y}{2} \end{aligned}\right. \tag{59}$$

For case distinctions with equation numbers, the package

```
\usepackage{cases}
```

provides by far the easiest solution:

```
\begin{numcases}{|x|=}
  x & for $x \geq 0$,
  \\
  -x & for $x < 0$.
\end{numcases}
```

$$|x| = \begin{cases} x & \text{for } x \geq 0, \qquad (60) \\ -x & \text{for } x < 0. \qquad (61) \end{cases}$$

Note the differences to the usual `cases`-environment:

- The left-hand side must be typeset as compulsory argument to the environment.
- The second column is not in math-mode but directly in text-mode.

For subnumbering we can use the corresponding `subnumcases`-environment:

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

$$P_U(u) = \begin{cases} 0.1 & \text{if } u = 0, \quad (62a) \\ 0.3 & \text{if } u = 1, \quad (62b) \\ 0.6 & \text{if } u = 2. \quad (62c) \end{cases}$$

6.3 Grouping numbered equations with a bracket

Sometimes, one would like to group several equations together with a bracket. We have already seen in Section 6.1 how this can be achieved by using `IEEEeqnarraybox` inside of an `equation`-environment:

```
\begin{equation}
  \left\{
    \begin{IEEEeqnarraybox}[
      \IEEEeqnarraystrutmode
      \IEEEeqnarraystrutsizeadd{2pt}
      {2pt}
      ][c]{rCl}
      \dot{x} & = & f(x,u)
      \\
      x+\dot{x} & = & h(x)
    \end{IEEEeqnarraybox}
  \right.
\end{equation}
```

$$\begin{cases} \dot{x} = f(x,u) \\ x + \dot{x} = h(x) \end{cases} \tag{63}$$

The problem here is that since the equation number is provided by the `equation`-environment, we only get one equation number. But here, a number for each equation would make much more sense.

We could again rely on `numcases` (see Section 6.2), but then we have no way of aligning the equations horizontally:

```
\begin{numcases}{}
  \dot{x} = f(x,u)
  \\
  x+\dot{x} = h(x)
\end{numcases}
```

$$\begin{cases} \dot{x} = f(x,u) & (64) \\ x + \dot{x} = h(x) & (65) \end{cases}$$

Note that misusing the second column of `numcases` is not an option either:

```
\ldots very poor typesetting:
\begin{numcases}{}
  \dot{x} & $\displaystyle
  = f(x,u)$
  \\
  x+\dot{x} & $\displaystyle
  = h(x)$
\end{numcases}
```

... very poor typesetting:

$$\begin{cases} \dot{x} & = f(x,u) & (66) \\ x + \dot{x} & = h(x) & (67) \end{cases}$$

The problem can be solved using `IEEEeqnarray`: we define an extra column on the most left that will only contain the bracket. However, as this bracket needs to be far higher than the line where it is defined, the trick is to use `\smash` to make its true height invisible to `IEEEeqnarray`, and then "design" its height manually using the `\IEEEstrut`-command. The number of necessary *jots* depends on the height of the equation and needs to be adapted manually:

```
\begin{IEEEeqnarray}{rrCl}
  & \dot{x} & = & f(x,u)
  \\*
  \smash{\left\{
      \IEEEstrut[8\jot]
    \right.}
  & x+\dot{x} & = & h(x)
  \\*
  & x+\ddot{x} & = & g(x)
\end{IEEEeqnarray}
```

$$\begin{cases} \dot{x} = f(x,u) & (68)\\ x+\dot{x} = h(x) & (69)\\ x+\ddot{x} = g(x) & (70) \end{cases}$$

The star in `\\*` is used to prevent the possibility of a page break within the structure.

This works fine as long as the number of equations is odd and the total height of the equations above the middle row is about the same as the total height of the equations below. For example, for five equations (this time using subnumbers for a change):

```
\begin{IEEEeqnarray}{rrCl}
  \IEEEyesnumber\IEEEyessubnumber*
  & a_1 + a_2 & = & f(x,u)
  \\*
  & a_1 & = & \frac{1}{2}h(x)
  \\*
  \smash{\left\{
      \IEEEstrut[16\jot]
    \right.}
  & b & = & g(x,u)
  \\*
  & y_{\theta} & = &
  \frac{h(x)}{10}
  \\*
  & b^2 + a_2 & = & g(x,u)
\end{IEEEeqnarray}
```

$$\begin{cases} a_1 + a_2 = f(x,u) & (71a)\\ a_1 = \frac{1}{2}h(x) & (71b)\\ b = g(x,u) & (71c)\\ y_\theta = \frac{h(x)}{10} & (71d)\\ b^2 + a_2 = g(x,u) & (71e) \end{cases}$$

However, if the heights of the equations differ greatly or if the number of equations is even, we get into a problem:

```
Bad example:
\begin{IEEEeqnarray}{rrCl}
  & a_1 + a_2 & = &
  \sum_{k=1}^{\frac{M}{2}} f_k(x,u)
  \label{eq:uneven1}
  \\*
  \smash{\left\{
      \IEEEstrut[15\jot]
    \right.}
  & b & = & g(x,u)
  \label{eq:uneven2}
  \\*
  & y_{\theta} & = & h(x)
  \label{eq:uneven3}
\end{IEEEeqnarray}
```

Bad example:

$$\begin{cases} a_1 + a_2 = \sum_{k=1}^{\frac{M}{2}} f_k(x,u) & (72)\\ b = g(x,u) & (73)\\ y_\theta = h(x) & (74) \end{cases}$$

or

```
Bad example:
\begin{IEEEeqnarray}{rrCl}
  & \dot{x} & = & f(x,u)
  \\*
  \smash{\left\{
      \IEEEstrut[8\jot]
    \right.} \nonumber
  \\*
  & y_{\theta} & = & h(x)
\end{IEEEeqnarray}
```

Bad example:

$$\begin{cases} \dot{x} = f(x,u) & (75)\\ y_\theta = h(x) & (76) \end{cases}$$

To solve this issue, we need manual tinkering. The basic idea is to use a hidden row at a place of our choice. To make the row hidden, we need to manually move the row above the hidden row down (respectively, the hidden row up!) and the row below up, by about half the usual line spacing each:

```
\begin{IEEEeqnarray}{rrCl}
  & \dot{x} & = & f(x,u)
  \\*[-0.625\normalbaselineskip]
% start invisible row
  \smash{\left\{
      \IEEEstrut[6\jot]
    \right.} \nonumber
% end invisible row
  \\*[-0.625\normalbaselineskip]
  & x+\dot{x} & = & h(x)
\end{IEEEeqnarray}
```

$$\begin{cases} \dot{x} = f(x,u) & (77)\\ x + \dot{x} = h(x) & (78) \end{cases}$$

In the case of an odd, but unequally sized number of equations, we can put the bracket on an individual row anywhere and then moving it up or down depending on how we need it. The example (72)–(74) with the three unequally sized equations now looks as follows:

```
\begin{IEEEeqnarray}{rrCl}
  & a_1 + a_2 & = &
  \sum_{k=1}^{\frac{M}{2}} f_k(x,u)
  \\*[-0.1\normalbaselineskip]
  \smash{\left\{
      \IEEEstrut[12\jot]
    \right.} \nonumber
  \\*[-0.525\normalbaselineskip]
  & b & = & g(x,u)
  \\*
  & y_{\theta} & = & h(x)
\end{IEEEeqnarray}
```

$$\left\{\begin{IEEEeqnarray}{rrCl} & a_1 + a_2 & = & \sum_{k=1}^{\frac{M}{2}} f_k(x,u) \quad (79)\\ & b & = & g(x,u) \quad (80)\\ & y_\theta & = & h(x) \quad (81)\end{IEEEeqnarray}\right.$$

Note how we can move the bracket up and down by changing the amount of shift in both `\\*[...\normalbaselineskip]`-commands: if we add $+2$ to the first and -2 to the second command (which makes sure that in total we have added $2-2=0$), we get:

```
\begin{IEEEeqnarray}{rrCl}
  & a_1 + a_2 & = &
  \sum_{k=1}^{\frac{M}{2}} f_k(x,u)
  \\*[1.9\normalbaselineskip]
  \smash{\left\{
      \IEEEstrut[12\jot]
    \right.} \nonumber
  \\*[-2.525\normalbaselineskip]
  & b & = & g(x,u)
  \\*
  & y_{\theta} & = & h(x)
\end{IEEEeqnarray}
```

$$\left\{\begin{IEEEeqnarray}{rrCl} & a_1 + a_2 & = & \sum_{k=1}^{\frac{M}{2}} f_k(x,u) \quad (82)\\ & b & = & g(x,u) \quad (83)\\ & y_\theta & = & h(x) \quad (84)\end{IEEEeqnarray}\right.$$

6.4 Matrices

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

```
\begin{equation}
  \mathsf{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}
```

$$\mathsf{P} = \begin{pmatrix} p_{11} & p_{12} & \ldots & p_{1n} \\ p_{21} & p_{22} & \ldots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \ldots & p_{mn} \end{pmatrix} \quad (85)$$

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).

6.5 Adapting the size of brackets

LaTeX offers the functionality of brackets being automatically adapted to the size of the expression they embrace. This is done using the pair of directives[14] `\left` and `\right`:

```
\begin{equation}
  a = \log \left( 1 +
    \sum_{k=1}^n b_k \right)
\end{equation}
```

$$a = \log\left(1 + \sum_{k=1}^{n} b_k\right) \tag{86}$$

The brackets do not need to be round, but can be of various types, e.g.,

```
\begin{equation*}
  \left\| \left(
    \left[ \left\{ \left|
      \left\lfloor \left\lceil
        \frac{1}{2}
      \right\rceil \right\rfloor
    \right| \right\} \right]
  \right) \right\|
\end{equation*}
```

$$\left\|\left(\left[\left\{\left|\left\lfloor\left\lceil\frac{1}{2}\right\rceil\right\rfloor\right|\right\}\right]\right)\right\|$$

It is important to note that `\left` and `\right` always must occur as a pair, but — as we have just seen — they can be nested. Moreover, the brackets do not need to match:

```
\begin{equation*}
  \left( \frac{1}{2}, 1 \right]
  \subset \mathbb{R}
\end{equation*}
```

$$\left(\frac{1}{2}, 1\right] \subset \mathbb{R}$$

One side can even be made invisible by using a dot instead of a bracket (`\left.` or `\right.`). We have already seen such examples in (57) or (59).

For an additional element in between a `\left`-`\right` pair that should have the same size as the surrounding brackets, the command `\middle` is available:

```
\begin{equation}
  H\left(X \, \middle| \,
    \frac{Y}{X} \right)
\end{equation}
```

$$H\left(X \,\middle|\, \frac{Y}{X}\right) \tag{87}$$

[14] Unfortunately, the `left`/`right` command pair has a weakness: in certain situations the chosen bracket size is slightly too big. For example, if expressions with larger superscripts like $a^{(1)}$ are typeset in displaystyle or, like here, in footnotes, we get $\left(a^{(1)}\right)$. I suggest to choose the bracket size manually in these cases: $\big(a^{(1)}\big)$.

Here both the size of the vertical bar and of the round brackets are adapted according to the size of $\frac{Y}{X}$.

Unfortunately, `\left-\right` pairing cannot be done across a line-break. So, if we wrap an equation using `multline` or `IEEEeqnarray`, we cannot have a `\left` on one side and the corresponding `\right` on the other side of the `\\`. In a first attempt, we might try to fix this by introducing a `\right.` on the left of `\\` and a `\left.` on the right of `\\` as shown in the following example:

```
\begin{IEEEeqnarray}{rCl}
  a & = & \log \left( 1 \right.
  \nonumber\\
  && \qquad \left. + \:
    \frac{b}{2} \right)
  \label{eq:wrong_try}
\end{IEEEeqnarray}
```

$$a = \log (1 \\ + \frac{b}{2}\Big) \tag{88}$$

As can be seen from this example, this approach usually does not work, because the sizes of the opening and closing brackets do not match anymore. In the example (88), the opening bracket adapts its size to "1", while the closing bracket adapts its size to $\frac{b}{2}$.

There are two ways to try to fix this. The by far easier way is to choose the bracket size manually:

```
\begin{IEEEeqnarray}{rCl}
  a & = & \log \bigg( 1
  \nonumber\\
  && \qquad +\:
  \frac{b}{2} \bigg)
\end{IEEEeqnarray}
```

$$a = \log \Big(1 \\ + \frac{b}{2}\Big) \tag{89}$$

There are four sizes available: in increasing order `\big`, `\Big`, `\bigg`, and `\Bigg`. This manual approach will fail, though, if the expression in the brackets requires a bracket size larger than `\Bigg`, as shown in the following example:

```
\begin{IEEEeqnarray}{rCl}
  a & = & \log \Bigg( 1
  \nonumber\\
  && \qquad + \sum_{k=1}^n
  \frac{e^{1+\frac{b_k^2}{c_k^2}}}
  {1+\frac{b_k^2}{c_k^2}}
  \Bigg) \label{eq:sizecorr1}
\end{IEEEeqnarray}
```

$$a = \log \Bigg(1 \\ + \sum_{k=1}^{n} \frac{e^{1+\frac{b_k^2}{c_k^2}}}{1+\frac{b_k^2}{c_k^2}}\Bigg) \tag{90}$$

For this case we need a trick: since we want to rely on a

```
\left( ... \right. \\ \left. ... \right)
```

construction, we need to make sure that both pairs are adapted to the same size. To that goal we define the following command in the document header:

```
\newcommand{\sizecorr}[1]{\makebox[0cm]{\phantom{$\displaystyle #1$}}}
```

We then pick the larger of the two expressions on either side of `\\` (in (90) this is the term on the second line) and typeset it a second time also on the other side of the line-break (inside of the corresponding `\left`-`\right` pair). However, since we do not actually want to see this expression there, we put it into `\sizecorr{}` and thereby make it both invisible and of zero width. In the example (90) this looks as follows:

```
\begin{IEEEeqnarray}{rCl}
  a & = & \log \left(
% copy-paste from below, invisible
  \sizecorr{
    \sum_{k=1}^n
    \frac{e^{1+\frac{b_k^2}{c_k^2}}}
    {1+\frac{b_k^2}{c_k^2}}
  }
% end copy-paste
  1 \right. \nonumber\\
  && \qquad \left. + \sum_{k=1}^n
  \frac{e^{1+\frac{b_k^2}{c_k^2}}}
  {1+\frac{b_k^2}{c_k^2}}
  \right) \label{eq:sizecorr2}
\end{IEEEeqnarray}
```

$$a = \log \left(1 + \sum_{k=1}^{n} \frac{e^{1+\frac{b_k^2}{c_k^2}}}{1+\frac{b_k^2}{c_k^2}} \right) \tag{91}$$

Note how the expression inside of `\sizecorr{}` does not actually appear, but is used for the computation of the correct bracket size.

6.6 Framed equations

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

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

$$\boxed{a = b + c} \tag{92}$$

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}
```

$$\boxed{a = b + c} \tag{93}$$

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}
```

$$\boxed{a = b + c} \tag{94}$$

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}
```

$$\boxed{\begin{IEEEeqnarraybox}{rCl} a &=& b + c \\ &=& d + e + f + g + h + i + j + k \\ && + l + m + n + o \\ &=& p + q + r + s \end{IEEEeqnarraybox}} \tag{95}$$

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}
```

$$\boxed{\begin{aligned} a &= b + c + d + e + f + g + h \\ &\quad + i + j + k + l + m + n \\ &\quad + o + p + q \end{aligned}} \tag{96}$$

in constrast 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}
```

$$\boxed{\begin{aligned} a &= b + c + d + e + f + g + h \\ &\quad + i + j + k + l + m + n \\ &\quad + o + p + q \end{aligned}} \tag{97}$$

- When changing the `IEEEeqnarray` into a `IEEEeqnarraybox`, be careful to delete any remaining `\nonumber` or `\IEEEnonumber` 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}
```

$$\boxed{\begin{aligned} a &= b + c + d + e + f + g + h \\ &\quad + i + j + k + l \end{aligned}}$$

6.7 Fancy frames

Fancier frames can be produced using the `framed` and the `pstricks` packages.[15] 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}
```

$$a = b + c \tag{98}$$
$$= d + e \tag{99}$$

Again, to ease one's life, we suggest to define this fancy box in the document header:

```
\newenvironment{whitebox}[1][0pt]{%
  \def\FrameCommand{\psshadowbox[shadowsize=0.3em, framesep=1.0em,%
    fillstyle=solid, fillcolor=white]}%
  \MakeFramed{\advance\hsize-\width \advance\hsize-#1 \FrameRestore}}%
{\endMakeFramed}
```

which then allows to typeset much more easily:

```
\begin{whitebox}
  \begin{IEEEeqnarray}{rCl}
    a & = & b + c
    \\
    & = & d + e
  \end{IEEEeqnarray}
\end{whitebox}
```

$$a = b + c \tag{100}$$
$$= d + e \tag{101}$$

Instead of a white frame, one could also define a shaded version (again, to be put into the document header):

```
\newgray{mygray}{0.9}
\newenvironment{graybox}[1][0pt]{%
  \def\FrameCommand{\psshadowbox[shadowsize=0.3em, framesep=1.0em,%
    fillstyle=solid, fillcolor=mygray]}%
  \MakeFramed{\advance\hsize-\width \advance\hsize-#1 \FrameRestore}}%
{\endMakeFramed}
```

[15] This will only work with pdflatex if one relies on the package `pst-pdf`. For more info we refer to the manual of `pst-pdf`.

These frames can also be put around larger structures like theorems:

```
\begin{graybox}
  \begin{theorem}
    This is a fancy theorem:
    we know by now that
    \begin{equation}
      a = b + c.
    \end{equation}
  \end{theorem}
\end{graybox}
```

Theorem 1. *This is a fancy theorem: we know by now that*

$$a = b + c. \tag{102}$$

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

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

6.8 Putting the QED correctly: proof

The package `amsthm` that we have used in Section 6.7 to generate a theorem actually also defines a `proof`-environment:

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

Proof. This is the proof of some theorem. Once the proof is finished a white box is put 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:

$$a = b + c. \qquad \square$$

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:

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

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:

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

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 only be needed on the last line for putting the `\qedhere`-command. 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:

$$a = b + c. \tag{103}$$ □

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:

$$a = b + c. \tag{104}$$

□

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:

$$\begin{aligned} a &= b + c && (105)\\ &= d + e. && (106) \end{aligned}$$

□

which contrasts with the poorer version:

```
\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:

$$\begin{aligned} a &= b + c && (107)\\ &= d + e. && (108) \end{aligned}$$

□

Note that `equation` does not handle the `\qedhere`-command correctly in all cases. Compare the following:

```
\begin{proof}
  This is a bad example for the
  usage of \verb+\qedhere+ in
  combination with \verb+equation+:
  \begin{equation}
    a = \sum_{\substack{x_i\\
        |x_i|>0}} f(x_i). \qedhere
  \end{equation}
\end{proof}
```

Proof. This is a bad example for the usage of `\qedhere` in combination with `equation`:

$$a = \sum_{\substack{x_i\\ |x_i|>0}} f(x_i). \tag{109}$$

□

with the much better solution:

```
\begin{proof}
  This is the corrected example
  using \verb+IEEEeqnarray+:
  \begin{IEEEeqnarray}{c+x*}
    a = \sum_{\substack{x_i\\
        |x_i|>0}} f(x_i).
    \\ & \qedhere\nonumber
  \end{IEEEeqnarray}
\end{proof}
```

Proof. This is the corrected example using `IEEEeqnarray`:

$$a = \sum_{\substack{x_i \\ |x_i|>0}} f(x_i). \tag{110}$$

□

You notice how the □ in the bad example is far too close the equation number and is actually inside the mathematical expression. A similar problem also occurs in the case of no equation number.

Hence, we recommend to use `\qedhere` exclusively in the combination with `IEEEeqnarray` and not with `equation`.

6.9 Putting the QED correctly: IEEEproof

`IEEEtrantools` also provides its own proof-environment that is slightly more flexible than the `proof` of `amsthm`: `IEEEproof`. Note that under the `IEEEtran`-class, `amsthm` is not permitted and one must use `IEEEproof`.

`IEEEproof` offers the command `\IEEEQEDhere` that produces the QED-symbol right at the place where it is invoked and will switch off the QED-symbol at the end.

```
\begin{IEEEproof}
  This is a short proof:
  \begin{IEEEeqnarray}{rCl+x*}
    a & = & b + c \\
    & = & d+ e \label{eq:qed}
    \\ &&& \nonumber\IEEEQEDhere
  \end{IEEEeqnarray}
\end{IEEEproof}
```

Proof: This is a short proof:

$$a = b + c \tag{111}$$
$$= d + e \tag{112}$$

■

So, in this sense `\IEEEQEDhere` plays the same role for `IEEEproof` as `\qedhere` for `proof`. Note, however, that their behavior is not completely equivalent: `\IEEEQEDhere` always puts the QED-symbol *right at the pace* it is invoked and does, e.g., not move it to the end of the line inside of a list or an `equation*`. So, for example, inside of a list an additional `\hfill` is needed:

```
\begin{IEEEproof}
  A proof containing a list and
  two QED-symbols:
  \begin{enumerate}
  \item Fact one.\IEEEQEDhere
  \item Fact two.\hfill\IEEEQEDhere
  \end{enumerate}
\end{IEEEproof}
```

Proof: A proof containing a list and two QED-symbols:

1. Fact one.■
2. Fact two. ■

Unfortunately, `\hfill` will not work inside an `equation`. To get the behavior of `\qedhere` there, one needs to use `\IEEEQEDhereeqn` instead:

```
\begin{IEEEproof}
  Placed directly behind math:
  \begin{equation*}
    a = b + c. \hfill\IEEEQEDhere
  \end{equation*}
  Moved to the end of line:
  \begin{equation*}
    a = b + c. \IEEEQEDhereeqn
  \end{equation*}
\end{IEEEproof}
```

Proof: Placed directly behind math:

$$a = b + c.\blacksquare$$

Moved to the end of line:

$$a = b + c. \qquad \blacksquare$$

`\IEEEQEDhereeqn` even works in situations with equation numbers, however, in contrast to `\qedhere` it does not move the QED-symbol to the next line, but puts it in front of the number:

```
\begin{IEEEproof}
  Placed directly before the
  equation number:
  \begin{equation}
    a = b + c. \IEEEQEDhereeqn
  \end{equation}
  With some additional spacing:
  \begin{equation}
    a = b + c. \IEEEQEDhereeqn\;
  \end{equation}
\end{IEEEproof}
```

Proof: Placed directly before the equation number:

$$a = b + c. \qquad \blacksquare(113)$$

With some additional spacing:

$$a = b + c. \qquad \blacksquare\ (114)$$

(To get the behavior where the QED-symbol is moved to the next line, use the approach based on `IEEEeqnarray` as shown in (112).)

Furthermore, `IEEEproof` offers the command `\IEEEQEDoff` to suppress the QED-symbol completely; it allows to change the QED-symbol to be an open box as in Section 6.8; and it allows to adapt the indentation of the proof header (default value is `2\parindent`). The latter two features are shown in the following example:

```
\renewcommand{\IEEEproofindentspace}{0em}
\renewcommand{\IEEEQED}{\IEEEQEDopen}
\begin{IEEEproof}
  Proof without
  indentation and an
  open QED-symbol.
\end{IEEEproof}
```

Proof: Proof without indentation and an open QED-symbol. □

The default QED-symbol can be reactivated again by redefining `\IEEEQED` to be `\IEEEQEDclosed`.

We end this section by pointing out that IEEE standards do not allow a QED-symbol and an equation put onto the same line. Instead one should follow the example (112).

6.10 Double-column equations in a two-column layout

Many scientific publications are in a two-column layout in order to save space. This means that the available width for the equations is considerably smaller than for a one-column layout and will cause correspondingly more line-breaks, in which case the advantages of the `IEEEeqnarray`-environment are even more pronounced.

However, there are very rare situations when the breaking of an equation into two or more lines will result in a very poor typesetting, even if `IEEEeqnarray` with all its tricks is used. In such a case, a possible solution is to span an equation over both columns. But the reader be warned:

> Unless there is no other solution, we strongly discourage from the usage of double-column equations in a two-column layout for aesthetic reasons and because the LaTeX code is rather ugly!

The trick is to use the `figure`-environment to create a floating object containing the equation similarly to a included graphic. Concretely, we have to use `figure*` to create a float that stretches over both columns. Since in this way the object becomes floating, unfortunately, the equation numbering does not work properly anymore and has to be done manually. We explain the details using an example. We start by defining the floating equation:

```
\newcounter{tempequationcounter}
\begin{figure*}[!t]
  \normalsize
  \setcounter{tempequationcounter}{\value{equation}}
  \begin{IEEEeqnarray}{rCl}
    \setcounter{equation}{115}
    a & = & b + c + d + e + f + g + h + i + j + k
    + l + m + n + o + p
    \nonumber \\
    && +\: q + r + s + t + u + v + w + x + y + z
    + \alpha + \beta + \gamma + \delta + \epsilon
    \label{eq:floatingequation}
  \end{IEEEeqnarray}
  \setcounter{equation}{\value{tempequationcounter}}
  \hrulefill
  \vspace*{4pt}
\end{figure*}
```

The exact location of this definition depends strongly on where the floating structure should be placed, i.e., it might need to be placed quite far away from the place where the equation is referred to in the text. Note that this might need quite some trial and error, particularly if there are other floating objects around to be placed by LaTeX.

The reference in the text will then look as follows:

$$a = b + c + d + e + f + g + h + i + j + k + l + m + n + o + p + q + r + s + t + u + v + w + x + y + z + \alpha + \beta + \gamma + \delta + \epsilon \quad (115)$$

```
\ldots and $a$ is given in
\eqref{eq:floatingequation}%
\addtocounter{equation}{1}
on the top of this page/on top of
page~\pageref{eq:floatingequation}.
```

> ... and a is given in (115) on the top of this page/on top of page 36.

A couple of explanations:

- We need to define an auxiliary equation counter `tempequationcounter` that will temporarily store the current value of the equation numbers at the moment when the floating equation is defined. Note that the definition

  ```
  \newcounter{tempequationcounter}
  ```

  should only be stated once per document. So if there are several floating equations, it must not be redefined again.

- The equation number of the floating equation **must be set manually**! That is, after we have finished typesetting the document, we check the value that the equation should have if it were typeset inside the text in a normal fashion. This number `xx` is then put into the command `\setcounter{equation}{xx}` inside the definition of the floating equation.

- The reference in the text must contain the command

  ```
  \addtocounter{equation}{1}
  ```

  that makes sure that the equation numbering is increased by one at this place. This way, inside of the text, the equation numbers will jump over one number, which is the number used by the floating equation.

- The reference "on the top of this page" must also be adapted manually, depending on where the equation actually appears!

- Note that due to a limitation of LaTeX, double-column floating objects cannot be placed at the bottom of pages, i.e., `\begin{figure*}[!b]` will not work correctly. This can be corrected if we include the following line in the header of our document:

  ```
  \usepackage{stfloats}
  ```

  However, this package is very invasive and might cause troubles with other packages. In particular, it cannot be used together with the package

```
\usepackage{fixltx2e}
```

which corrects another bug of LaTeX, namely to make sure that the order of single- and double-column floating objects are kept in sequence.

7 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 an `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 LaTeX code:

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

$$\begin{aligned} a &= b + c && (116)\\ &= d + e + f + g + h + i + j + k + l + m + n + o && (117) \end{aligned}$$

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}
```

$$\begin{aligned} a &= b + c && (118)\\ &= d + e + f + g + h + i + j + k \\ &\quad + l + m + n + o && (119) \end{aligned}$$

- **Control-c n**: As **Control-c b**, but without adding the additional space `\:`.
- **Control-c Control-b**: Remove a line-break (undo of **Control-c b** and **Control-c n**). Position the cursor before the `\nonumber` and press **Control-c Control-b**.
- **Control-c m**: Insert a `\IEEEeqnarraymulticol`-command. This is very helpful when the LHS is too long. Suppose you have typed the following LaTeX code:

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

$$a+b+c+d+e+f+g+h+i+j = k+l \tag{120}$$
$$= m+n \tag{121}$$

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

$$a+b+c+d+e+f+g+h+i+j$$
$$= k+l \tag{122}$$
$$= m+n \tag{123}$$

- Finally, in the `dot_emacs`-file settings are given that make `IEEEeqnarray` and `IEEEeqnarraybox` known to Emacs' LaTeX-mode, reftex, and ispell. This way many standard Emacs commands can be used as usual also in the context of `IEEEeqnarray`. For example, **Control-c (** will add an equation label.

8 Some Final Remarks and Acknowledgments

The "rules" stated in this document are purely based on my own experience with typesetting LaTeX in my publications and on my — some people might say unfortunate — habit of incorporating many 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 LaTeX or PDF file! As a matter of fact, any kind of feedback, criticism, suggestion, etc., is highly appreciated! Write to

`stefan.moser@ieee.org`

Thanks!

I would like to mention that during the writing and updating of this document I profited tremendously from the help of Michael Shell, the author of `IEEEtran`. He was always available for explanations when I got stuck somewhere. Moreover, I gratefully acknowledge the comments from (in alphabetical order) Helmut Bölcskei, Amos Lapidoth, Edward Ross, Omar Scaglione and Sergio Verdú.

Stefan M. Moser