How to Typeset Equations in LaTeX

Stefan M. Moser

April 19, 2024
Version 5.0

Contents

Chapter 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 worse 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.

The most current version of this manual can be found at

https://moser-isi.ethz.ch/

1.1 Reading Guide

Over the years this manual has grown to quite an extended size. If you have limited time, read Section 3.1 about the basic use of `IEEEeqnarray`. If you have little more time, Chapters 3 and 4 contain more details of how to use the environment. These two chapters cover most cases that occur in practice.

Chapter 5 discusses some more advanced problems and their solutions. The interested reader can find details on the issues with traditional typesetting tools in Chapter 2.

Finally, Chapter 6 contains some hints and tricks about the editor Emacs, and Chapter 7 presents some more goodies and tricks for nice typesetting with LaTeX.

1.2 Notation

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.

1.3 Included Files

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 Chapter 6).
- `IEEEtrantools.sty` [2015/08/26 V1.5 by Michael Shell]: package needed for the `IEEEeqnarray`-environment.
- `IEEEtran.cls` [2015/08/26 V1.8b by Michael Shell]: LaTeX document class package for papers in IEEE format.
- `IEEEtran_HOWTO.pdf` [2015/08]: official manual of the `IEEEtran`-class. The part about `IEEEeqnarray` is found in Appendix F.

Note that `IEEEtran.cls` and `IEEEtrantools.sty` are provided automatically by any up-to-date LaTeX-distribution and are included only for the sake of completeness.

1.4 LaTeX-Setup

The strength of LaTeX concerning typesetting of mathematics is strongly based on the package `amsmath`. Every current distribution of LaTeX will come with this package included, so you only need to make sure that the following line is included in the preamble of your document:

```
\usepackage{amsmath}
```

In this manual, we propose the usage of the `IEEEeqnarray`-environment, which is provided by the package[1] `IEEEtrantools`. Thus, one needs to include the following line in the preamble of your document:

```
\usepackage{IEEEtrantools}
```

This manual is based on the current version 1.5 of `IEEEtrantools` (respectively, version 1.8b of `IEEEtran.cls`). Note that this version was published in 2015, so almost any LaTeX installation will be up-to-date.[2]

Throughout this document it is assumed that both `amsmath` and `IEEEtrantools` are loaded.

[1] This package is also distributed together with this manual, but it is already included in any regular LaTeX distribution. Note that if a document uses the `IEEEtran`-class, then `IEEEtrantools` is loaded automatically and must not be included separately.

[2] You can check the version on your system using `kpsewhich IEEEtrantools.sty` to find the path to the used file and then viewing it.

Chapter 2

Issues with Traditional Commands

In this section we will point out the drawbacks of the traditional commands used to typeset equations in LaTeX. This will serve as a motivation for our proposed approach based on `IEEEeqnarray`.

2.1 $$, \[, displaymath

The most obvious drawback of the commands

`$$...$$`
`\[...\]`
`displaymath`

is the lack of equation-numbering. This alone is already reason enough to completely avoid them!

But in addition to that, the vertical spacing around the equation is wrong with `$$...$$`, and because this is plain TeX syntax, the command cannot be modified.[3] The other two commands are redefined by `amsmath` to be synonymous to `equation*`, thus their spacing is correct. Nevertheless, we strongly discourage the use of them, particularly of the former because it additionally also results in a very poorly readable source code.

In summary:

> **NEVER** ever use the `$$...$$`-command!

(and do not use `\[...\]` either...)

[3] The same could be said for the inline-math command `$...$` that is also plain TeX syntax and should actually be replaced by `\(...\)`. But apart from a slightly worse error handling, currently there is no disadvantage in using the traditional `$`.

2.2 equation

A much better approach is to rely on 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*}
```

$$a = b + c$$

Unfortunately, `equation` uses an automatic mechanism to move the equation number onto the next line if the expression is too long. While this is convenient, sometimes the equation number is forced onto the next line, even if there was still enough space available on the line:

```
\begin{equation}
  a = \sum_{k=1}^n\sum_{\ell=1}^n
  \sin \bigl(2\pi \, b_k \,
  c_{\ell} \, d_k \, e_{\ell} \,
  f_k \, g_{\ell} \, h_k \, xy
  \bigr)
\end{equation}
```

$$a = \sum_{k=1}^{n}\sum_{\ell=1}^{n} \sin\bigl(2\pi \, b_k \, c_\ell \, d_k \, e_\ell \, f_k \, g_\ell \, h_k \, xy\bigr) \tag{2}$$

With `IEEEeqnarray`, the placement of the equation number is fully under our control and we can decide whether we prefer it to be on the same line or on the next:[4]

```
\begin{IEEEeqnarray}{c}
  a = \sum_{k=1}^n\sum_{\ell=1}^n
  \sin \bigl(2\pi \, b_k \,
  c_{\ell} \, d_k \, e_{\ell} \,
  f_k \, g_{\ell} \, h_k \, xy
  \bigr)
  \IEEEeqnarraynumspace
\end{IEEEeqnarray}
```

$$a = \sum_{k=1}^{n}\sum_{\ell=1}^{n} \sin\bigl(2\pi \, b_k \, c_\ell \, d_k \, e_\ell \, f_k \, g_\ell \, h_k \, xy\bigr) \tag{3}$$

or

```
\begin{IEEEeqnarray}{c}
  a = \sum_{k=1}^n\sum_{\ell=1}^n
  \sin \bigl(2\pi \, b_k \,
  c_{\ell} \, d_k \, e_{\ell} \,
  f_k \, g_{\ell} \, h_k \, xy
  \bigr)
  \nonumber\\*
\end{IEEEeqnarray}
```

$$a = \sum_{k=1}^{n}\sum_{\ell=1}^{n} \sin\bigl(2\pi \, b_k \, c_\ell \, d_k \, e_\ell \, f_k \, g_\ell \, h_k \, xy\bigr) \tag{4}$$

[4] For a detailed explanation of the syntax of this command, see Chapter 3.

2.3 multline

If an equation is too long, we have to wrap it somehow. A way to achieve this is the `multline`-environment provided by the amsmath-package:

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

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

The difference to the `equation`-environment is that an arbitrary line-break (or also multiple line-breaks) can be introduced. One simply places a \\ at the location where the equation should be wrapped.

Similarly to `equation*` there also exists a `multline*`-version for preventing an equation number.

However, in spite of its ease of use, `multline` has many issues. First of all, the indentations on the following lines are rather heuristic and usually not in accordance with the mathematical expression:

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

$$a+b+c+d+e+f+g+h+i \\ = j+k+l+m+n \\ = j+k+l+m+n \\ = j+k+l+m+n \quad (6)$$

We see that the first two equality-signs can be considered reasonable, but the third definitely is completely out of place. Also note that it is not possible to add additional equation numbers for the first two equalities. Thus, we see that `multline` can only handle a single equation.

Unfortunately, a single equation is usually not handled correctly either. Consider the following common situation:

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

$$a = b+c+d+e+f+g+h+i+j+k+l+m+n+o+p \quad (7)$$

Obviously, the RHS is too long to fit on one line. So, we add a line-break:

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

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

This is of course much better than (7), but it has the disadvantage that the equality sign loses its natural stronger importance over the plus operator in front of k. A much better solution is provided by the `IEEEeqnarray`-environment:[5]

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

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

Note how the wrapped part of the expression on the RHS is aligned with 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
\end{multline}
```

$$\begin{aligned} &a + b + c + d + e + f + g + h \\ &= i + j + k + l + m + n + o + p + q + r + s \end{aligned} \quad (10)$$

Again this looks much better using `IEEEeqnarray`:

```
\begin{IEEEeqnarray}{rCl}
  \IEEEeqnarraymulticol{3}{l}{%
    a + b + c + d + e + f + g + h
  }\nonumber\\*%
  & = & i + j + k + l + m + n + o
  + p + q + r + s \nonumber\\*
\end{IEEEeqnarray}
```

$$\begin{aligned} &a + b + c + d + e + f + g + h \\ &= i + j + k + l + m + n + o + p + q + r + s \end{aligned} \quad (11)$$

For more details see Section 4.2.

Finally, there is the drawback that `multline` allows for an equation to start right on top of a page. This usually does not look good, and `equation` and `IEEEeqnarray` both try to put a line of text first before the equation starts.

Over the years, we have only found the following two situations where the use of `multline` might make sense.

[5] Again, the details on its usage will be discussed in Chapter 3.

2.3.1 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}
  a + b + c + d + e + f \\
  + g + h + i + j + k + l \\
  + m + n + o + p + q
\end{multline}
```

$$a+b+c+d+e+f \\ +g+h+i+j+k+l \\ +m+n+o+p+q \quad (12)$$

For a way of achieving the same result with `IEEEeqnarray`, see Section 4.6.

2.3.2 Case 2: 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 a good solution:

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

$$a+b+c+d+e+f+g \\ +h+i+j+k+l=m \quad (13)$$

Other possible solutions for such a situation are presented in Section 4.2.

2.4 align

To group multiple equations, the `align`-environment could be used:

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

$$a = b+c \quad (14)$$
$$= d+e \quad (15)$$
$$= f+g \quad (16)$$

While this looks neat as long as every equation fits onto one line, this approach does not work anymore 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}
```

$$a = b+c \quad (17)$$
$$= d+e+f+g+h+i+j+k+l$$
$$+m+n+o \quad (18)$$
$$= p+q+r+s \quad (19)$$

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

What is needed is a vertical structure that distinguishes LHS, equality-sign, and RHS of the equation:

```
\begin{IEEEeqnarray}{rCl}
  a & = & b + c
  \label{eq:threecolumns1}
  \\
  & = & d + e + f + g + h + i
  + j + k + l \nonumber\\
  && +\> m + n + o
  \\
  & = & p + q + r + s
  \label{eq:threecolumns3}
\end{IEEEeqnarray}
```

$$\begin{aligned} a &= b + c && (20)\\ &= d + e + f + g + h + i + j + k + l \\ &\quad + m + n + o && (21)\\ &= p + q + r + s && (22) \end{aligned}$$

Note the three corresponding columns that are defined using `{rCl}`. For more details, see Section 3.1.

Note that the `align`-environment can also be used to group several blocks of equations beside each other:

```
\begin{align*}
  a & = b + c  &  x & = y + z \\
    & = d + e  &    & = u + w
\end{align*}
```

$$\begin{aligned} a &= b + c & x &= y + z \\ &= d + e & &= u + w \end{aligned}$$

However, also in this situation, we recommend to use the `IEEEeqnarray`-environment defining six rather than four columns:

```
\begin{IEEEeqnarray*}{rCl+rCl}
  a & = & b + c  & x & = & y + z \\
    & = & d + e  &   & = & u + w
\end{IEEEeqnarray*}
```

$$\begin{aligned} a &= b + c & x &= y + z \\ &= d + e & &= u + w \end{aligned}$$

The `+` denotes a stretchable space between two columns. Note that this approach also allows for line-breaks within the equations:

```
\begin{IEEEeqnarray*}{rCl+rCl}
  a & = & b + c &
  x & = & y + z + t + s \\
  &&&& & \> + m + n \\
  & = & d + e  &
  & = & u + w
\end{IEEEeqnarray*}
```

$$\begin{aligned} a &= b + c & x &= y + z + t + s \\ & & &\quad + m + n \\ &= d + e & &= u + w \end{aligned}$$

2.5 eqnarray

The `eqnarray`-environment officially provides a three-column structure as we have seen in (20)–(22):

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

$$\begin{aligned} a &= b+c && (23)\\ &= d+e+f+g+h+i+j+k+l \\ &\quad +m+n+o && (24)\\ &= p+q+r+s && (25) \end{aligned}$$

Note, however, that `eqnarray` is not an `amsmath`-command[6] and has a few *very severe* disadvantages:

- The spaces around the equality sign in the middle column are far too big and do not match the regular spacing:

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

  $$a = a = a \qquad (26)$$

- The RHS 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}
  ```

  $$\begin{aligned} a &= b+c && (27)\\ &= d+e+f+g+h^2+i^2+j && (28) \end{aligned}$$

- The `eqnarray`-environment offers a command `\lefteqn{...}` that is supposed to 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}
  ```

  $$\begin{aligned} &a+b+c+d+e+f+g+h \\ &\quad = i+j+k+l+m && (29)\\ &\quad = n+o+p+q+r+s && (30) \end{aligned}$$

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

[6] It stems from the dawn of LaTeX.

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

$$\begin{aligned} & a+b+c+d+e+f+g+h \\ &= \quad i+j \end{aligned} \tag{31}$$

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

In summary:

NEVER ever use the `eqnarray`-environment!

Chapter 3

IEEEeqnarray

Please recall that the package `IEEEtrantools` needs to be included for `IEEEeqnarray` to be defined. See Section 1.4.

3.1 Basic Usage

The main strength of `IEEEeqnarray` is that it requires an additional argument that defines the *columns* of the equation array. If we typeset a single equation, this argument will be `{c}` specifying a single column that is centered:

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

$$a = b + c + d + e \tag{32}$$

As usual, the starred version suppresses the equation number:

```
\begin{IEEEeqnarray*}{c}
  a = b + c + d + e
\end{IEEEeqnarray*}
```

$$a = b + c + d + e$$

In the situation when we have a sequence of several equalities or an expression that needs to be wrapped, usually `{rCl}` will be specified, defining three columns:

- `r`: the first column right-justified, containing the LHS of the equation;
- `C`: the middle one centered with a little more space around it (therefore we specify capital `C` instead of lower-case `c`), containing the equality-signs;
- `l`: and the third column left-justified, containing the RHS of the equation.

For example:

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

$$\begin{aligned} a &= b + c && (33)\\ &= d + e && (34)\\ &= f + g && (35) \end{aligned}$$

Note how &-signs are used to separate the columns in each line and how \\ starts a new line. If some column is not filled on a particular line, the corresponding &-sign is stated immediately:

```
\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}
```

$$\begin{aligned} a &= b + c && (36)\\ &= d + e + f + g + h + i + j + k \\ &\quad + l + m + n + o && (37)\\ &= p + q + r + s && (38) \end{aligned}$$

So, here on the fifth line, the && means that we jump to the third column immediately (the "$+\, l$" is aligned directly below "d").

3.2 Column Types

There exist the following column-types:

type	description
l	left-flushed math entry
c	centered math entry
r	right-flushed math entry
L	left-flushed math entry with more spacing around
C	centered math entry with more spacing around
R	right-flushed math entry with more spacing around
s	left-flushed text entry
t	centered text entry
u	right-flushed text entry

Besides, additional spacing can be added by , and : and ; and ' and " in increasing order. As an example, consider the following array with an additional fourth column (in text-mode) for explanations:[7]

```
\begin{IEEEeqnarray}{rCl"s}
  a & = & b + c & (by Theorem~1)
  \\
  & = & d + e & (algebraic transf.)
\end{IEEEeqnarray}
```

$$\begin{aligned} a &= b + c && \text{(by Theorem 1)} && (39)\\ &= d + e && \text{(algebraic transf.)} && (40) \end{aligned}$$

[7] For more examples of spacing, we refer to Section 5.2. More spacing types can be found in the examples given in Sections 4.3 and 5.9, and in the official manual `IEEEtran_HOWTO.pdf`, which is distributed together with this short introduction (the part about `IEEEeqnarray` can be found in Appendix F).

3.3 Rules on How to Wrap Equations

Unfortunately, wrapped equations are usually less easy to read than not-wrapped ones. To improve the readability, one should follow 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.

3.4 Some Settings of IEEEeqnarray

Even though we recommend to use `IEEEeqnarray` exclusively for all situations, a mixed use in combination with `equation` and/or `align` is in principle possible. But there are two issues that one needs to be aware of where `IEEEeqnarray` behaves differently from the other commands. Luckily, this is not a problem because `IEEEeqnarray` allows for very simple adaptation of its behavior.

3.4.1 Vertical Spacing

The vertical spacing of `IEEEeqnarray` and of `equation`/`align` are not exactly identical, even though the difference is small. In general, `IEEEeqnarray` has the tendency to squeeze the equations together a tiny bit more. This can be adapted by the following command that needs to be put into the preamble:

```
\renewcommand*{\IEEEeqnarraydecl}{\setlength{\jot}{1.2\IEEEnormaljot}}
```

The default value is `1.0\IEEEnormaljot`. By increasing it, more space is added in between lines of `IEEEeqnarray`.

3.4.2 Font of Equation Number

In `equation` the equation number is typeset in regular font even if the equation is within an environment[8] of different font:

[8] A typical example of such a situation is an equation inside of a theorem that is typeset in italic font.

```
\textbf{\textit{
    This is our main result:
  \begin{equation}
    a = b + c
  \end{equation}
  in a bold-italic environment.}}
```

This is our main result:

$$a = b + c \tag{41}$$

in a bold-italic environment.

Note that the equation number is neither bold nor italic. In contrast, `IEEEeqnarray` respects the settings of the environment:

```
\textbf{\textit{
    This is our main result:
  \begin{IEEEeqnarray}{c}
    a = b + c
  \end{IEEEeqnarray}
  in a bold-italic environment.}}
```

This is our main result:

$$a = b + c \tag{42}$$

in a bold-italic environment.

If this behavior of `IEEEeqnarray` is undesired, it can be changed:[9]

```
\renewcommand{\theequationdis}{{\normalfont (\theequation)}}
\renewcommand{\theIEEEsubequationdis}{{\normalfont (\theIEEEsubequation)}}
```

Now, `IEEEeqnarray` behaves like `equation`:

```
\textbf{\textit{
    This is our main result:
  \begin{IEEEeqnarray}{rCl}
    a & = & b + c \\
    & = & d + e \IEEEyesnumber
    \IEEEyessubnumber
  \end{IEEEeqnarray}
  in a bold-italic environment.}}
```

This is our main result:

$$\begin{aligned} a &= b + c && (43)\\ &= d + e && (44a) \end{aligned}$$

in a bold-italic environment.

Note, however, that coloring will be taken over by both environments:

```
\textbf{\textit{\color{red}
    This is our main result:
  \begin{equation}
    a = b + c
  \end{equation}
  in a bold-italic environment.}}
```

This is our main result:

$$a = b + c \tag{45}$$

in a bold-italic environment.

In the case of `IEEEeqnarray`, this can be changed easily in the same manner:

```
\renewcommand{\theequationdis}{{\normalfont\color{black} (\theequation)}}
```

```
\textbf{\textit{\color{red}
    This is our main result:
  \begin{IEEEeqnarray}{c}
    a = b + c
  \end{IEEEeqnarray}
  in a bold-italic environment.}}
```

This is our main result:

$$a = b + c \tag{46}$$

in a bold-italic environment.

[9] For an explanation of the subnumbering, see Section 4.4.

Chapter 4

More Details about IEEEeqnarray

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

4.1 Shift to the Left: IEEEeqnarraynumspace

If a line overlaps with the equation number as in (28), the command

```
\IEEEeqnarraynumspace
```

can be used to fix things. It is added to the line that is too long and makes sure that the whole equation array is shifted to the left by the amount corresponding to the size of the equation number (i.e., 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 + m
  \\
  & = & l + n + o
\end{IEEEeqnarray}
```

we get

$$\begin{aligned} a &= b + c && (47)\\ &= d + e + f + g + h + i + j + k + m && (48)\\ &= l + n + o && (49) \end{aligned}$$

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

$$\begin{aligned} a &= b + c && (50)\\ &= d + e + f + g + h + i + j + k + m && (51)\\ &= l + n + o && (52) \end{aligned}$$

Note that if there is not enough space on the line, this shift will force the numbers to cross the right boundary of the text. So be sure to check the result!

```
The boundary of the text can be
seen from this text above the
equation array. The number is
clearly beyond it:
\begin{IEEEeqnarray}{rCl}
  a & = & d + e + f + g + h
  + i + j + k + l + m + n
  \IEEEeqnarraynumspace
\end{IEEEeqnarray}
```

The boundary of the text can be seen from this text above the equation array. The number is clearly beyond it:

$$a = d + e + f + g + h + i + j + k + l + m + n \tag{53}$$

In such a case one needs to wrap the equation somewhere.

4.2 LHS is too Long: IEEEeqnarraymulticol

If the LHS is too long, `IEEEeqnarray` offers the `\IEEEeqnarraymulticol`-command (which is a correct implementation of the faulty `\lefteqn`-command):

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

$$\begin{aligned} a + b + c + d + e + f + g + h & \\ &= i + j && (54)\\ &= k + l + m && (55) \end{aligned}$$

The usage is identical to the `\multicolumns`-command in the `tabular`-environment. The first argument `{3}` specifies that three cells shall be combined into one, which — as specified by the second argument `{l}` — will be left-justified. The third argument contains then the contents of the combined cell. Note that the `\IEEEeqnarraymulticol`-command can be used in general within `IEEEeqnarray` to combine cells together.

Then we add a line-break, with the additional command `\nonumber` to prevent a number for this "half-line". Note that we recommend to add a `*` to the line-break `\\` to prevent a page-break at this position (for improved readability).

Finally, the `\quad` will shift the equality sign slightly to the right to make sure that the LHS is left of the RHS. Note that by adapting this distance one can easily adapt the depth of the equality signs,[10] e.g.,

[10] One quad is the distance that looks good in most cases.

```
\begin{IEEEeqnarray}{rCl}
  \IEEEeqnarraymulticol{3}{l}{
    a + b + c + d + e + f
    + g + h
  }\nonumber\\* \qquad\qquad
  & = & i + j
  \label{eq:label45}
  \\
  & = & k + l + m
\end{IEEEeqnarray}
```

$$\begin{aligned} a+b+c+d+e+f+g+h & \\ &= i+j && (56)\\ &= k+l+m && (57) \end{aligned}$$

Warning: `\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 of `IEEEeqnarray`, which is OK in general, but will result in an error if it is followed by the `\IEEEeqnarraymulticol`-command. Thus:

> We strongly recommend to put each label at the end of the corresponding equation just before the line-break \\ (or the end of the equation array).

4.3 Line-Break: Unary versus Binary Operators

If an equation is split onto 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}
```

$$\begin{aligned} a &= b+c && (58)\\ &= d+e+f+g+h+i+j+k & \\ &\quad +l+m+n+o && (59)\\ &= p+q+r+s && (60) \end{aligned}$$

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}
```

$$\begin{aligned} a &= b+c && (61)\\ &= d+e+f+g+h+i+j+k & \\ &\quad +l+m+n+o && (62)\\ &= p+q+r+s && (63) \end{aligned}$$

Compare the space between + and l.

Warning: The distinction between the *unary operator* (sign) and the *binary operator* (addition/subtraction) is not satisfactorily solved in LaTeX.[11] 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 brackets with fixed size like `(`, `\bigl(`, or `\bigr)`).

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*}
```

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

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 (62)), the only option is to put in the correct space `\>` manually.[12]

In the following example, compare 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*}
```

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

[11] The problem actually goes back to TeX.

[12] The space command `\>` is like `\,` but with a certain flexibility of being stretchable. It is defined as `medmuskip = 4mu plus 2mu minus 4mu`.

We learn:

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

4.4 Equation Numbering

4.4.1 Numbers and Subnumbers

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 until the end of the `IEEEeqnarray`-environment or until another starred command is invoked.

Consider the following extensive example, where we are going to use the index of the variable b_i as "line number":

```
\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*}
(\ldots some text\ldots)
\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}
(\ldots more text\ldots)
\begin{IEEEeqnarray}{rCl}
  \IEEEyesnumber\label{eq:block}
  \IEEEyessubnumber*
  a
  & = & b_{24} \\
  & = & b_{25} \label{eq:subeq_b}\\
  & = & b_{26}
\end{IEEEeqnarray}
```

$$
\begin{aligned}
a &= b_{1} \\
&= b_{2} && (64)\\
&= b_{3} \\
&= b_{4} && (65)\\
&= b_{5} && (66)\\
&= b_{6} && (67)\\
&= b_{7} \\
&= b_{8} && (68)\\
&= b_{9} \\
&= b_{10} \\
&= b_{11} && (68a)\\
&= b_{12} && (68b)\\
&= b_{13} && (69)\\
&= b_{14} && (69a)\\
&= b_{15} && (69b)
\end{aligned}
$$

(…some text…)

$$
\begin{aligned}
a &= b_{16} && (69c)\\
&= b_{17} && (69d)\\
&= b_{18} && (70a)\\
&= b_{19} && (70b)\\
&= b_{20} && (71)\\
&= b_{21} && (72)\\
&= b_{22} \\
&= b_{23} && (73)
\end{aligned}
$$

(…more text…)

$$
\begin{aligned}
a &= b_{24} && (74a)\\
&= b_{25} && (74b)\\
&= b_{26} && (74c)
\end{aligned}
$$

Note that the behavior on line 13 (i.e., the line containing b_{13}) is probably unwanted: there the command `\IEEEyesnumber` temporarily switches to a normal equation number (and thereby 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 starting directly with a new subnumber is shown on line 18 and on line 24. Also note that the subnumbering is not reset by the end of an `IEEEeqnarray`-environment, as can be seen on line 16. The reset only happens on line 18 because of the command `\IEEEyesnumber`.

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 referring to an equation with a subnumber must always be given *after* the `\IEEEyessubnumber` command. Otherwise the label will refer to the current (or future) main number, which is usually undesired. E.g., the label `eq:bad_placement` in line 16 points[13] (wrongly) to (70).

A correct example is shown in (74) and (74b): the label `\label{eq:block}` refers to the whole block, and the label `\label{eq:subeq_b}` refers to the corresponding subequation.

We learn once again:

> A label should always be put at the end of the equation it belongs to (i.e., right in front of the line-break \\).

Besides preventing unwanted results, this rules also increases the readability of the source code and prevents a compilation error in the situation of an `\IEEEeqnarraymulticol`-command after a label-definition (see Section 4.2).

4.4.2 Hyperlinks

As this document demonstrates, hyperlinking works (almost) seamlessly with `IEEEeqnarray`. For this document we simply included

```
\usepackage[colorlinks=true,linkcolor=blue]{hyperref}
```

in the preamble, and then all references automatically become hyperlinks.

There is only one small issue that you might have noticed already: the reference (74) points into nirvana. The reason for this is that there is no actual equation number (74) generated and therefore `hyperref` does not create the corresponding hyperlink. This can be fixed, but requires some more advanced LaTeX-programming. Copy-paste the following code into the document preamble (or your stylefile):

[13] To understand this, note that when the `label`-command 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 18 where the equation counter is incremented for the first time.

```
\makeatletter
\def\IEEElabelanchoreqn#1{\bgroup
\def\@currentlabel{\p@equation\theequation}\relax
\def\@currentHref{\@IEEEtheHrefequation}\label{#1}\relax
\Hy@raisedlink{\hyper@anchorstart{\@currentHref}}\relax
\Hy@raisedlink{\hyper@anchorend}\egroup}
\makeatother
\newcommand{\subnumberinglabel}[1]{\IEEEyesnumber
  \IEEEyessubnumber*\IEEElabelanchoreqn{#1}}
```

Now, `\IEEElabelanchoreqn{...}` creates an anchor for a hyperlink to an invisible equation number. The command `\subnumberinglabel` then sets this anchor and at the same time activates subnumbering, simplifying our typesetting:

```
We have
\begin{IEEEeqnarray}{rCl}
  \subnumberinglabel{eq:block2}
  a & = & b + c
  \label{eq:block2_eq1}\\
  & = & d + e
  \label{eq:block2_eq2}
\end{IEEEeqnarray}
and
\begin{IEEEeqnarray}{c}
  \IEEEyessubnumber*
  f = g - h + i
  \label{eq:block2_eq3}
\end{IEEEeqnarray}
```

We have

$$a = b + c \tag{75a}$$
$$= d + e \tag{75b}$$

and

$$f = g - h + i \tag{75c}$$

Now (75) refers to the whole block (the hyperlink points to the first line of the first equation array), and (75a), (75b), and (75c) point to the corresponding subequations.

4.4.3 Alternative Subnumbers: subequations

We conclude this section by remarking that `IEEEeqnarray` is fully compatible with the `subequations`-environment. Thus, (75) can also be created in the following way:

```
We have
\begin{subequations}
  \label{eq:block2_alt}
  \begin{IEEEeqnarray}{rCl}
    a & = & b + c
    \label{eq:block2_eq1_alt}\\
    & = & d + e
    \label{eq:block2_eq2_alt}
  \end{IEEEeqnarray}
  and
  \begin{IEEEeqnarray}{c}
    f = g - h + i
    \label{eq:block2_eq3_alt}
  \end{IEEEeqnarray}
\end{subequations}
```

We have

$$a = b + c \tag{76a}$$
$$= d + e \tag{76b}$$

and

$$f = g - h + i \tag{76c}$$

Note, however, that the hyperlink of (76) points to the beginning of the `subequations`-environment and not onto the first line of the equation array as in (75)!

4.5 Page-Breaks within IEEEeqnarray

By default, `amsmath` does not allow page-breaks within multiple equations, and this setting is taken over by `IEEEeqnarray`, too. Usually, however, this is too restrictive, particularly, if a document contains long equation arrays. Luckily, this behavior can be adapted by putting the following line into the document preamble:

```
\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; and a value 10'000 basically prevents page-breaks (which is the default given in `amsmath`).

We recommend to use a value 1000 that in principle allows page-breaks, but still asks LaTeX to check if there is no better way.

4.6 Emulating multline

Since we propose to rely on `IEEEeqnarray` exclusively, we should present ways on how to replace the other tools. For `equation` the replacement clearly is `IEEEeqnarray` with a column specification `{c}`; and for `align` we have proposed the column specification `{rCl}`. We now would like to propose a possible emulation of `multline`.

We do this by specifying only one column `{l}` and using `\IEEEeqnarraymulticol`[14] after the line-break(s) to adapt the column type of the new line(s). In addition, we manually add some shift:

```
\begin{IEEEeqnarray*}{l}
  a + b + c + d + e + f
  \\ \qquad
  +\> g + h + i + j + k + l
  \qquad \\
  \IEEEeqnarraymulticol{1}{r}{
    +\> m + n + o + p + q  }
  \IEEEyesnumber
\end{IEEEeqnarray*}
```

$$\begin{aligned} a+b+c+d+e+f \\ +g+h+i+j+k+l \\ +m+n+o+p+q \end{aligned} \tag{77}$$

[14] Compare with the explanations in Section 4.2.

Chapter 5

Advanced Typesetting

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

5.1 Alignment of Several Equation Arrays

Sometimes it looks elegant if one can align not just the equations within one array, but between several arrays (with regular text in between). This can be achieved by actually creating one single large array and adding additional text in between. For example, (75) could be typeset as follows:

```
We have
\begin{IEEEeqnarray}{rCl}
  \subnumberinglabel{eq:block3}
  a & = & b + c
  \label{eq:block3_eq1}\\
  & = & d + e
  \label{eq:block3_eq2}\\
  \noalign{\noindent and
    \vspace{2\jot}}
  f & = & g - h + i
  \label{eq:block3_eq3}
\end{IEEEeqnarray}
```

We have

$$a = b + c \tag{78a}$$
$$= d + e \tag{78b}$$

and

$$f = g - h + i \tag{78c}$$

Note how the equality-sign in (78c) is aligned to the equality-signs of (78a) and (78b).

In the code, we add the text "and" into the array using the command `\noalign{...}` and then manually add some vertical spacing.

5.2 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).[15] Hence, `IEEEeqnarraybox` is a replacement both for `array` and `tabular`.

```
This is a silly table:
\begin{Center}
  \begin{IEEEeqnarraybox}{t.t.t}
    \textbf{Item} &
    \textbf{Color} &
    \textbf{Count} \\
    cars & green & 17 \\
    trucks & red & 4 \\
    bikes & blue & 25
  \end{IEEEeqnarraybox}
\end{Center}
```

This is a silly table:

Item	Color	Count
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* (see also the table in Section 3.2), `v` for a vertical line, and `V` for a vertical double-line.[16] More details can be found in Tables IV and V on page 18 in the manual `IEEEtran_HOWTO.pdf`.

Another example:[17]

```
\begin{IEEEeqnarray}{c}
  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{IEEEeqnarray}
```

$$P_U(u) = \begin{cases} 0.1 & \text{if } u = 0, \\ 0.3 & \text{if } u = 1, \\ 0.6 & \text{if } u = 2. \end{cases} \tag{79}$$

Here `?` is a larger 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 baseline. Default is `[b]`, i.e., if we do not specify this option, we get the following (in this case unwanted) result:

[15] 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.

[16] Note, however, that it is highly recommendable *not* to use any vertical lines in tables. See the discussion in Section 7.1.2.

[17] For another way of generating case distinctions, see Section 5.3.

```
\begin{IEEEeqnarray*}{c}
  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{IEEEeqnarray*}
```

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

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{IEEEeqnarray*}{c}
  \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{IEEEeqnarray*}
```

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{IEEEeqnarray*}{c}
  \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{IEEEeqnarray*}
```

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 an `\IEEEstrut`:

```
\begin{IEEEeqnarray*}{c}
  \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{IEEEeqnarray*}
```

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.

We end this section by emphasizing once again that usually a good table design will *not* make use of any vertical lines (see also the discussion in Section 7.1.2).

5.3 Case Distinctions

Case distinctions can be generated using `IEEEeqnarraybox` as shown in Section 5.2. However, in the standard situation the usage of `cases` (provided by `amsmath`) is simpler and we therefore recommend to use this:

```
\begin{IEEEeqnarray}{c}
  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{IEEEeqnarray}
```

$$P_U(u) = \begin{cases} 0.1 & \text{if } u = 0, \\ 0.3 & \text{if } u = 1, \\ 0.6 & \text{if } u = 2. \end{cases} \tag{80}$$

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

```
\begin{IEEEeqnarray}{c}
  \left.
  \begin{IEEEeqnarraybox}[\IEEEeqnarraystrutmode
    \IEEEeqnarraystrutsizeadd{2pt}{2pt}][c]{rCl}
    x & = & a + b\\
    y & = & a - b
  \end{IEEEeqnarraybox}
  \, \right\}  \iff  \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.
  \IEEEeqnarraynumspace
  \label{eq:example_left_right2}
\end{IEEEeqnarray}
```

$$\left.\begin{aligned} x &= a+b \\ y &= a-b \end{aligned}\right\} \iff \begin{cases} a = \dfrac{x}{2}+\dfrac{y}{2} \\ b = \dfrac{x}{2}-\dfrac{y}{2} \end{cases} \tag{81}$$

If we would like to have a distinct equation number for each case,[18] the package

```
\usepackage{cases}
```

provides by far the easiest solution:

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

$$|x| = \begin{cases} x & \text{for } x \geq 0, \qquad (82) \\ -x & \text{for } x < 0. \qquad (83) \end{cases}$$

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}
```

$$P_U(u) = \begin{cases} 0.1 & \text{if } u = 0, \qquad (84a) \\ 0.3 & \text{if } u = 1, \qquad (84b) \\ 0.6 & \text{if } u = 2. \qquad (84c) \end{cases}$$

[18] I presonally prefer a single number as in my understanding a case distinction is simply a more complicated form of a single expression.

5.4 Grouping Numbered Equations with a Bracket

Sometimes, one would like to group several equations together with a bracket. We have already seen in (81) how this can be achieved by using `IEEEeqnarraybox` inside of a regular `IEEEeqnarray`-environment:

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

$$\begin{cases} \dot{x} = f(x,u) \\ x+\dot{x} = h(x) \end{cases} \tag{85}$$

The problem here is that since the equation number is provided by the outer `IEEEeqnarray`-environment, we only get one equation number. But here in this context, an individual number for each equation would make much more sense.

We could again rely on `numcases` (see Section 5.3), but then we have no way of aligning the equations horizontally:

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

... poor typesetting:

$$\begin{cases} \dot{x} = f(x,u) & (86) \\ x+\dot{x} = h(x) & (87) \end{cases}$$

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:

$$\begin{cases} \dot{x} &= f(x,u) & (88) \\ x+\dot{x} &= h(x) & (89) \end{cases}$$

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[7\jot]
    \right.}
  & x+\dot{x} & = & h(x)
  \\*
  & x+\ddot{x} & = & g(x)
\end{IEEEeqnarray}
```

$$\begin{cases} \dot{x} = f(x,u) & (90)\\ x+\dot{x} = h(x) & (91)\\ x+\ddot{x} = g(x) & (92) \end{cases}$$

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}
  \subnumberinglabel{eq:block4}
  & a_1 + a_2 & = & f(x,u)
  \\*
  & a_1 & = & \frac{1}{2}h(x)
  \\*
  \smash{\left\{
      \IEEEstrut[14\jot]
    \right.}
  & b & = & g(x,u)
  \\*
  & y_{\theta} & = &
  \frac{h(x)}{10}
  \\*
  & b^2 + a_2 & = & g(x,u)
\end{IEEEeqnarray}
```

$$\begin{cases} a_1 + a_2 = f(x,u) & (93a)\\ a_1 = \frac{1}{2}h(x) & (93b)\\ b = g(x,u) & (93c)\\ y_\theta = \frac{h(x)}{10} & (93d)\\ b^2 + a_2 = g(x,u) & (93e) \end{cases}$$

However, we do get into problems if the heights of the equations differ greatly:

```
Bad example: uneven height
distribution:
\begin{IEEEeqnarray}{rrCl}
  \subnumberinglabel{eq:uneven}
  & a_1 + a_2 & = &
  \sum_{k=1}^{\frac{M}{2}} f_k(x,u)
  \\*
  \smash{\left\{
      \IEEEstrut[13\jot]
    \right.}
  & b & = & g(x,u)
  \\*
  & y_{\theta} & = & h(x)
\end{IEEEeqnarray}
```

Bad example: uneven height distribution:

$$\begin{cases} a_1 + a_2 = \sum_{k=1}^{\frac{M}{2}} f_k(x,u) & (94a)\\ b = g(x,u) & (94b)\\ y_\theta = h(x) & (94c) \end{cases}$$

or if the number of equations is even:

```
Another bad example:
even number of equations:
\begin{IEEEeqnarray}{rrCl}
  & \dot{x} & = & f(x,u)
  \\*
  \smash{\left\{
      \IEEEstrut[7\jot]
    \right.} \nonumber
  \\*
  & y_{\theta} & = & h(x)
\end{IEEEeqnarray}
```

Another bad example: even number of equations:

$$\begin{cases} \dot{x} = f(x,u) & (95) \\ y_\theta = h(x) & (96) \end{cases}$$

To solve this issue, we need manual tinkering. In the latter case, 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 down the row above the hidden row and to move up the row below, both by about half the usual line spacing:

```
\begin{IEEEeqnarray}{rrCl}
  & \dot{x} & = & f(x,u)
  \\*[-0.625\normalbaselineskip]
% start invisible row
  \smash{\left\{
      \IEEEstrut[5\jot]
    \right.} \nonumber
% end invisible row
  \\*[-0.625\normalbaselineskip]
  & x+\dot{x} & = & h(x)
\end{IEEEeqnarray}
```

$$\begin{cases} \dot{x} = f(x,u) & (97) \\ x + \dot{x} = h(x) & (98) \end{cases}$$

In the former case of unequally sized 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 (94) with the three unequally sized equations then looks as follows:

```
\begin{IEEEeqnarray}{rrCl}
  \subnumberinglabel{eq:uneven2}
  & a_1 + a_2 & = &
  \sum_{k=1}^{\frac{M}{2}} f_k(x,u)
  \\*[-0.1\normalbaselineskip]
  \smash{\left\{
      \IEEEstrut[10\jot]
    \right.} \nonumber
  \\*[-0.525\normalbaselineskip]
  & b & = & g(x,u)
  \\*
  & y_{\theta} & = & h(x)
\end{IEEEeqnarray}
```

$$\begin{cases} a_1 + a_2 = \sum_{k=1}^{\frac{M}{2}} f_k(x,u) & (99a) \\ b = g(x,u) & (99b) \\ y_\theta = h(x) & (99c) \end{cases}$$

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 obtain:

```
\begin{IEEEeqnarray}{rrCl}
  \subnumberinglabel{eq:uneven3}
  & a_1 + a_2 & = &
  \sum_{k=1}^{\frac{M}{2}} f_k(x,u)
  \\*[1.9\normalbaselineskip]
  \smash{\left\{
      \IEEEstrut[10\jot]
    \right.} \nonumber
  \\*[-2.525\normalbaselineskip]
  & b & = & g(x,u)
  \\*
  & y_{\theta} & = & h(x)
\end{IEEEeqnarray}
```

$$\begin{cases} a_1 + a_2 = \sum_{k=1}^{\frac{M}{2}} f_k(x,u) & \text{(100a)} \\ b = g(x,u) & \text{(100b)} \\ y_\theta = h(x) & \text{(100c)} \end{cases}$$

Warning: Note that size of one `\jot` is changed in the settings-command given in Section 3.4.1, which changes the vertical spacing of `IEEEeqnarray`. So, if we change this setting, we will have to manually fix the size of all brackets again!

5.5 Matrices

Matrices could be generated by `IEEEeqnarraybox`, however, the environment `pmatrix` and its siblings (all provided by `amsmath`) are easier to use:

```
\begin{IEEEeqnarray}{c}
  \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}
  \IEEEeqnarraynumspace
\end{IEEEeqnarray}
```

$$\mathsf{P} = \begin{pmatrix} p_{11} & p_{12} & \ldots & p_{1n} \\ p_{21} & p_{22} & \ldots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \ldots & p_{mn} \end{pmatrix} \qquad (101)$$

Note that it is not necessary to specify the number of columns (or rows) in advance. However, by default the number of columns is restricted to be at most 10. If one needs a matrix with more columns, the maximum number of columns has to be increased using the following command:

```
\setcounter{MaxMatrixCols}{13}
```

where in this example the maximum number of columns has been set to 13.

More possibilities are `bmatrix` (for matrices with square brackets), `Bmatrix` (curly brackets), `vmatrix` (|), `Vmatrix` (‖), and `matrix` (no brackets at all).

5.6 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 `\left` and `\right`:

```
\begin{IEEEeqnarray}{c}
  f \left( \sum_{k=1}^n b_k \right)
  =
  f \Biggl( \sum_{k=1}^n b_k \Biggr)
  \label{eq:adapt_bracket_size}
\end{IEEEeqnarray}
```

$$f\left(\sum_{k=1}^{n} b_k\right) = f\left(\sum_{k=1}^{n} b_k\right) \tag{102}$$

Unfortunately, the `\left`-`\right` pair has two weaknesses. First, it adds too much space before and after the brackets: in (102), compare the space between the f and the opening bracket on the LHS and on the RHS. This can easily be remedied by including the following two lines into the document preamble:

```
\usepackage{mleftright}
\mleftright
```

This fixes the spacing issue and the example (102) looks as follows:

```
\begin{IEEEeqnarray}{c}
  f \left( \sum_{k=1}^n b_k \right)
  =
  f \Biggl( \sum_{k=1}^n b_k \Biggr)
\end{IEEEeqnarray}
```

$$f\left(\sum_{k=1}^{n} b_k\right) = f\left(\sum_{k=1}^{n} b_k\right) \tag{103}$$

Second, in certain situations the chosen bracket size is too large. For example, this happens when expressions with large superscripts are typeset in a smaller font size like in the following footnote.[19] In this case it is easiest to adapt the bracket size manually.

5.6.1 General Usage

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

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

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

[19] In footnotes, we get $\left(a^{(1)}\right)$, which obviously is too big. I suggest to choose the bracket size manually using `bigl(` and `bigr)` in such a case: $\bigl(a^{(1)}\bigr)$.

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, and the brackets themselves do not need to match:

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

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

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 (79) or (81).

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

$$H\left(X \,\middle|\, \frac{Y}{X}\right) \tag{104}$$

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

5.6.2 Left-Right Pairs with Line-Breaks

Unfortunately, `\left`-`\right` pairing cannot be done across a line-break (or even only acress different cells in `IEEEeqnarray`). So, if we wrap an equation using `IEEEeqnarray`, we cannot have a `\left` before and the corresponding `\right` after a line-break `\\`. In a first attempt, we might try to fix this by introducing a `\right.` before the line-break and a `\left.` after the line-break, 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}
```

$$a = \log(1 \qquad\qquad + \frac{b}{2}\Big) \tag{105}$$

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 (105), 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 \biggl( 1
  \nonumber\\
  && \qquad +\>
  \frac{b}{2} \biggr)
\end{IEEEeqnarray}
```

$$a = \log\biggl(1 + \frac{b}{2}\biggr) \tag{106}$$

There are four sizes available: in increasing order `\bigl`, `\Bigl`, `\biggl`, and `\Biggl` (with the corresponding `..r`-versions).

This manual approach will fail, though, if the expression in the brackets requires a bracket size larger than `\Biggl`, as shown in the following example:

```
\begin{IEEEeqnarray}{rCl}
  a & = & \log \Biggl( 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}}
  \Biggr)
  \label{eq:sizecorr1}
\end{IEEEeqnarray}
```

$$a = \log\Biggl(1 + \sum_{k=1}^{n} \frac{e^{1+\frac{b_k^2}{c_k^2}}}{1+\frac{b_k^2}{c_k^2}}\Biggr) \tag{107}$$

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 preamble:

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

We then pick the larger of the two expressions on either side of `\\` (in (107) 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 (but correct height!). In the example (107) 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}
```

$$a = \log\left(1 + \sum_{k=1}^{n} \frac{e^{1+\frac{b_k^2}{c_k^2}}}{1+\frac{b_k^2}{c_k^2}}\right) \tag{108}$$

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

5.7 Framed Equations

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

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

$$\boxed{a = b + c} \tag{109}$$

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

$$\boxed{a = b + c} \tag{110}$$

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 preamble:

```
\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{IEEEeqnarray}{c}
  \eqbox{
    a = b + c
  }
\end{IEEEeqnarray}
```

$$\boxed{a = b + c} \tag{111}$$

Unfortunately, the `boxed{...}` command does not allow for line-breaks within its box. 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}
```

$$\boxed{\begin{aligned} a &= b + c \\ &= d + e + f + g + h + i + j + k \\ &\quad + l + m + n + o \\ &= p + q + r + s \end{aligned}} \tag{112}$$

where, for a change, we use `equation` as surrounding environment instead of our usual `IEEEeqnarray` with argument `{c}`.

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`. For example:

```
\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}
```

$$a = b + c + d + e + f + g + h \\ + i + j + k + l + m + n \\ + o + p + q \tag{113}$$

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}
```

$$a = b + c + d + e + f + g + h \\ + i + j + k + l + m + n \\ + o + p + q \tag{114}$$

- 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}
```

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

5.8 Fancy Frames

Fancier frames can be produced using the `mdframed` package. Use the following commands in the preamble of your document:[20]

[20] The `mdframed`-package should be loaded after amsthm.sty.

```
\usepackage{tikz}
\usetikzlibrary{shadows} %defines shadows
\usepackage[framemethod=tikz]{mdframed}
```

Then we can produce all kinds of fancy frames. We start by defining a certain style (still in the preamble of your document):

```
\global\mdfdefinestyle{myboxstyle}{%
  shadow=true,
  linecolor=black,
  shadowcolor=black,
  shadowsize=6pt,
  nobreak=false,
  innertopmargin=10pt,
  innerbottommargin=10pt,
  leftmargin=5pt,
  rightmargin=5pt,
  needspace=1cm,
  skipabove=10pt,
  skipbelow=15pt,
  middlelinewidth=1pt,
  afterlastframe={\vspace{5pt}},
  aftersingleframe={\vspace{5pt}},
  tikzsetting={%
    draw=black,
    very thick}
}
```

These settings are quite self-explanatory. Just play around! Now we define different types of framed boxes:

```
% framed box that allows page-breaks
\newmdenv[style=myboxstyle]{whitebox}
\newmdenv[style=myboxstyle,backgroundcolor=black!20]{graybox}

% framed box that CANNOT be broken at end of page
\newmdenv[style=myboxstyle,nobreak=true]{blockwhitebox}
\newmdenv[style=myboxstyle,backgroundcolor=black!20,nobreak=true]{blockgraybox}

% invisible box that CANNOT be broken at end of page
\newmdenv[nobreak=true,hidealllines=true]{blockbox}
```

As the name suggests, the `graybox` adds a gray background color into the box, while the background in `whitebox` remains white. Moreover, `blockwhitebox` creates the same framed box as `whitebox`, but makes sure that whole box is typeset onto one single page, while the regular `whitebox` can be split onto two (or even more) pages.

Examples:

```
\begin{whitebox}
  \begin{IEEEeqnarray}[
      \vspace{-0.5\baselineskip}
    ]{rCl}
    a & = & b + c
    \\
    & = & d + e
  \end{IEEEeqnarray}
\end{whitebox}
```

$$a = b + c \tag{115}$$
$$= d + e \tag{116}$$

and

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

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

$$a = b + c. \tag{117}$$

Note that in the former example, we have removed some space above the equation (that is automatically added by `IEEEeqnarray`) in order to have proper spacing. In the latter example we have assumed that the `theorem`-environment has been defined in the preamble:

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

5.9 Putting the QED Correctly: proof

The package `amsthm` that we have used in Section 5.8 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: (bad version)
  \begin{equation*}
    a = b + c.
  \end{equation*}
\end{proof}
```

Proof. This is a proof that ends with an equation: (bad version)

$$a = b + c.$$

□

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: (correct)
  \begin{equation*}
    a = b + c. \qedhere
  \end{equation*}
\end{proof}
```

Proof. This is a proof that ends with an equation: (correct)

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

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

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

Proof. This is a proof that ends with an equation array: (wrong)

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

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. Thereby, `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.

Luckily, there is a very simple remedy: We explicitly define these stretching columns ourselves!

```
\begin{proof}
  This is a proof that ends
  with an equation array: (correct)
  \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: (correct)

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

Here, the `+` in `{+rCl+x*}` denotes a stretchable space, 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 a numbered equation, we have a similar problem. If you compare

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

Proof. This is a proof that ends with a numbered equation: (bad)

$$a = b + c. \tag{118}$$

□

with

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

Proof. This is a proof that ends with a numbered equation: (better)

$$a = b + c. \tag{119}$$

□

you notice that in the (better) 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: (correct)
  \begin{IEEEeqnarray}{rCl+x*}
    a & = & b + c \\
    & = & d + e. \label{eq:star}
    \\* &&& \qedhere\nonumber
  \end{IEEEeqnarray}
\end{proof}
```

Proof. This is a proof that ends with an equation array: (correct)

$$\begin{aligned} a &= b + c && (120)\\ &= d + e. && (121) \end{aligned}$$

□

which contrasts with the poorer version:

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

Proof. This is a proof that ends with an equation array: (bad)

$$\begin{aligned} a &= b + c && (122)\\ &= d + e. && (123) \end{aligned}$$

□

Note that we use a starred line-break in (121) to prevent a page-break just before the QED-sign.

We would like to point out that `equation` does not handle the `\qedhere`-command correctly in all cases. Consider the following example:

```
\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`:

$$a = \sum_{\substack{x_i\\|x_i|>0}} f(x_i). \tag{124}$$

□

A much better solution can be achieved with `IEEEeqnarray`:

```
\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`:

$$a = \sum_{\substack{x_i\\|x_i|>0}} f(x_i). \tag{125}$$

□

Here, we add an additional line to the equation array without number, and put `\qedhere` command there (within the additional empty column on the right). Note 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 not to use `\qedhere` in combination with `equation`, but exclusively with `IEEEeqnarray` — or in short: use `IEEEeqnarray` only!

5.10 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 therefore `proof` is not defined, i.e., 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 with a
  correctly put end-of-proof sign:
  \begin{IEEEeqnarray}{rCl+x*}
    a & = & b + c \\
    & = & d+ e \label{eq:qed}
    \\* &&& \nonumber\IEEEQEDhere
  \end{IEEEeqnarray}
\end{IEEEproof}
```

Proof: This is a short proof with a correctly put end-of-proof sign:

$$a = b + c \tag{126}$$
$$= d + e \tag{127}$$

■

So, in this sense `\IEEEQEDhere` plays the same role for `IEEEproof` as `\qedhere` for `proof`. Note, however, that their behavior is not exactly equivalent: `\IEEEQEDhere` always puts the QED-symbol *right at the place* it is invoked and does not move it to the end of the line. 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. ■

5.10.1 IEEEproof and equation

We strongly discourage the use of `IEEEproof` in combination with `equation`, but for the sake of completeness we summarize its use here anyway.

The main issue is that `\hfill` will not work inside an `equation`. Thus, one must use `\IEEEQEDhereeqn` instead:

```
\begin{IEEEproof}
  hfill does not work (very bad):
  \begin{equation*}
    a = b + c. \hfill\IEEEQEDhere
  \end{equation*}
  With IEEEQEDhereeqn things work
  again as before:
  \begin{equation*}
    a = b + c. \IEEEQEDhereeqn
  \end{equation*}
\end{IEEEproof}
```

Proof: hfill does not work (very bad):

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

With IEEEQEDhereeqn things work again as before:

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

But, identically to `\qedhere` and the `proof`-environment, this approach is not satisfactory once the expression has a larger height and uses space above and below the current line:

```
\begin{IEEEproof}
  Pretty ugly:
  \begin{equation*}
    a = b + \sum_{k=1}^{2^n} x_k.
    \IEEEQEDhereeqn
  \end{equation*}
\end{IEEEproof}
```

Proof: Pretty ugly:

$$a = b + \sum_{k=1}^{2^n} x_k. \qquad \blacksquare$$

To add insult to injury, `\IEEEQEDhereeqn` could even be used in situations with equation numbers:

```
\begin{IEEEproof}
  Most horrible typesetting:
  \begin{equation}
    a = b + c. \IEEEQEDhereeqn
  \end{equation}
\end{IEEEproof}
```

Proof: Most horrible typesetting:

$$a = b + c. \qquad \blacksquare(128)$$

To get the behavior where the QED-symbol is moved to the next line, use the approach based on `IEEEeqnarray` as shown in (127).

To summarize:

> Do *not* use `equation` with `\IEEEQEDhere` and `\IEEEQEDhereeqn`, but rely exclusively on `IEEEeqnarray` and `\IEEEQEDhere`.

5.10.2 Settings of IEEEproof

`IEEEproof` offers the following settings:

- The command `\IEEEQEDoff` is used suppress the QED-symbol completely.
- By redefining `\IEEEQED` to `\IEEEQEDopen` we can change the QED-symbol to an open box (instead of the filled black box defined by `\IEEEQEDclosed`), similarly to the style of `proof` shown in Section 5.9.
- The indentation of the proof header can be adapted (the default value is rather large: `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. □

5.11 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 in a one-column layout and will cause correspondingly more line-breaks. In such a setting, the advantages of the `IEEEeqnarray`-environment are even more pronounced.

Nevertheless, there are 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 use of double-column equations in a two-column layout for aesthetic reasons and because the corresponding 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, the equation numbering has to be carefully taken care of.

We explain the details using an example. We start by defining two auxiliary equation counters:

```
\newcounter{storeeqcounter}
\newcounter{tempeqcounter}
```

The counter `storeeqcounter` will store the equation number that is assigned to the floating equation, and the counter `tempeqcounter` will be used to restore the equation counter to the correct number after it was temporarily set to the floating equation's number stored in `storeeqcounter`.

Note that if there are several floating equations in a document, each needs its own unique definition of a `storeeqcounter`, i.e., one needs to introduce different names for these counters (e.g., `storeeqcounter_one`, `storeeqcounter_two`, etc.). The counter `tempeqcounter` can be reused for all floating equations.

Now, in the text where we refer to the floating equation, we need to make sure that the equation number is increased by one (i.e., at this place the equation numbering will jump over one number, which is the number assigned to the floating equation), and then we need to store this number for later use. This looks as follows:

```
\ldots and $a$ is given in
\eqref{eq:floatingeq}
%
%% Increase current equation
%% number and store it:
\addtocounter{equation}{1}%
\setcounter{storeeqcounter}%
 {\value{equation}}%
%
on the top of this page/on top of
Page~\pageref{eq:floatingeq}.
```

…and a is given in (129) on the top of this page/on top of Page 49.

Note that one must manually adapt the LaTeX code to either the phrase "on the top of this page" or the phrase "on top of Page 49", depending on where the equation actually appears.

Finally we typeset the floating equation:

$$\begin{aligned} a &= b + c + d + e + f + g + h + i + j + k + l + m + n + o + p \\ &\quad + q + r + s + t + u + v + w + x + y + z + \alpha + \beta + \gamma + \delta + \epsilon \end{aligned} \tag{129}$$

```
\begin{figure*}[!t]
  \normalsize
  \setcounter{tempeqcounter}{\value{equation}} % temp store of current value
  \begin{IEEEeqnarray}{rCl}
    \setcounter{equation}{\value{storeeqcounter}} % number of this equation
    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:floatingeq}
  \end{IEEEeqnarray}
  \setcounter{equation}{\value{tempeqcounter}} % restore correct value
  \hrulefill
  \vspace*{4pt}
\end{figure*}
```

The exact location of this definition depends strongly on where the floating structure should be appear, i.e., it might have to be placed quite far away from the text where the equation is referred to.[21] Note that this might need some trial and error, particularly if there are other floating objects around to be placed by LaTeX.

Be aware 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 preamble of our document:

```
\usepackage{stfloats}
```

However, this package is very invasive and might cause troubles with other packages.[22]

[21] It needs to be placed *after* the reference in the text, though, as otherwise the equation number stored in `storeeqcounter` is not defined yet. This could again be fixed, but only if we set the equation number (i.e., `storeeqcounter`) manually (ugly!!).

[22] In particular, it cannot be used together with the package `fixltx2e.sty`. Luckily, the latter is not needed anymore for TeXLive 2015 or newer.

Chapter 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 an `IEEEeqnarray`-environment with argument `{rCl}`. (This is similar to using **Control-c Control-e** to insert some general environment.)
- **Control-c I**: As **Control-c i**, but the *-version.
- **Control-c o**: Insert an `IEEEeqnarray`-environment with argument `{c}`.
- **Control-c O**: As **Control-c o**, 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}
```

$$\begin{aligned} a &= b + c && (130)\\ &= d + e + f + g + h + i + j + k + l + m + n + o && (131) \end{aligned}$$

  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}
```

$$\begin{aligned} a &= b + c && (132)\\ &= d + e + f + g + h + i + j + k \\ &\quad + l + m + n + o && (133) \end{aligned}$$

- **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}
```

$$a+b+c+d+e+f+g+h+i+j = k+l \tag{134}$$
$$= m+n \tag{135}$$

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}
```

$$a+b+c+d+e+f+g+h+i+j$$
$$= k+l \tag{136}$$
$$= m+n \tag{137}$$

- **Control-c s**: Insert a `\IEEEnumspace` command. Suppose you have typed the following LaTeX code:

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

$$a = b+c+d+e+f+g+h+i \tag{138}$$
$$= j+k+l \tag{139}$$

After compiling you realize that the equation needs to be shifted. You now just put the cursor on or before the \\-sign and type **Control-c s**. Then you get

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

$$a = b+c+d+e+f+g+h+i \tag{140}$$
$$= j+k+l \tag{141}$$

- **Control-c 9**: Insert a label. Simply put the cursor on or before the \\-sign and type **Control-c 9**. Then a label is generated and inserted.

- 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`.

Chapter 7

Some Useful Packages and Definitions

7.1 Useful Packages

The following packages are very useful and we recommend to have them always directly included.

7.1.1 ragged2e

The package `ragged2e` provides the environments `FlushLeft`, `FlughRight`, and `Center`, and the commands `\RaggedRight`, `\RaggedLeft`, and `\Centering`, which do exactly what you expect. They are a replacement for the corresponding commands in lower case and fix some serious issue that prevents LaTeX to hyphenate text within these environments. To demonstrate the difference, we present the following example:

```
\begin{flushleft}
  This fundamentally extensive
  construction of an environment
  consisting of neverending words
  eradicates the possibility of
  understanding the intentionally
  complicated language.
\end{flushleft}
```

This fundamentally extensive construction of an environment consisting of neverending words eradicates the possibility of understanding the intentionally complicated language.

```
\begin{FlushLeft}
  This fundamentally extensive
  construction of an environment
  consisting of neverending words
  eradicates the possibility of
  understanding the intentionally
  complicated language.
\end{FlushLeft}
```

This fundamentally extensive construction of an environment consisting of neverending words eradicates the possibility of understanding the intentionally complicated language.

Thus, always make sure to include

```
\usepackage{ragged2e}
```

in the document preamble and by default use the new commands with capital letters in their names.

7.1.2 array and booktabs

Even though we can use `IEEEeqnarraybox` for creating tables, it often is simpler to rely on `tabular`. One should, however, always include the packages `array` and `booktabs` that fix and improve the `tabular`-environment considerably:

```
\usepackage{array,booktabs}
```

In general, a professionally typeset table should not contain any vertical, but only horizontal lines. For this, the commands `\toprule`, `\midrule`, and `bottomrule` are provided. When defining cells, one has the options `l` (left aligned), `c` (centered), `r` (right aligned), and `p{width}` (fixed cell of given width). Moreover, we can use `>{...}` to include commands that are part of each cell, and with `@{...}` we provide input to be put *between* cells.

As an example, consider the following table with three columns: the first is left aligned, the other two have a fixed width. Moroever, we add `\RaggedRight` to each cell in columns two and three:

```
\begin{Center}
\begin{tabular}{@{}l
  >{\RaggedRight}p{0.45\textwidth}
  >{\RaggedRight}p{0.4\textwidth}
  @{}}
  \toprule
  & \textbf{Menu}
  & \textbf{Comments}
  \\\midrule
  1 & salad with French dressing
  & starter; what drinks shall
    we offer?
  \\
  2 & steak and noodles, with
  mushroom sauce, green
  vegetables
  & standard, but tasty; wine?
  \\
  3 & apple and strawberry ice cream
  & dessert for after lunch
  \\\bottomrule
\end{tabular}
\end{Center}
```

	Menu	Comments
1	salad with French dressing	starter; what drinks shall we offer?
2	steak and noodles, with mushroom sauce, green vegetables	standard, but tasty; wine?
3	apple and strawberry ice cream	dessert for after lunch

Note that `@{}` at both sides are used to remove the default space that is added at the boundaries of the table. To demonstrate this effect, we typeset the same table a second time without removing the space on the left and adding exclamation marks on the right:

```
\begin{Center}
\begin{tabular}{l
  >{\RaggedRight}p{0.45\textwidth}
  >{\RaggedRight}p{0.4\textwidth}
  @{!}}
  \toprule
  & \textbf{Menu}
  & \textbf{Comments}
  \\\midrule
  1 & salad with French dressing
  & starter; what drinks shall
    we offer?
  \\
  2 & steak and noodles, with
  mushroom sauce, green
  vegetables
  & standard, but tasty; wine?
  \\
  3 & apple and strawberry ice cream
  & dessert for after lunch
  \\\bottomrule
\end{tabular}
\end{Center}
```

	Menu	Comments	!
1	salad with French dressing	starter; what drinks shall we offer?	!
2	steak and noodles, with mushroom sauce, green vegetables	standard, but tasty; wine?	!
3	apple and strawberry ice cream	dessert for after lunch	!

In summary:

> Do not use vertical lines in tables!

7.1.3 csquotes

The package

```
\usepackage[autostyle,german=swiss]{csquotes}
```

provides the command `\enquote{...}` and `\foreignquote{...}` to put expressions into quotes. The choice of quotes depends on the chosen language of the document (the option `autostyle` checks the language provided by `babel`). And for some languages, some style-variants can be defined (like the Swiss version of quotes for German used in the example here).

Example:

```
\begin{FlushLeft}
  He claimed: \enquote{Quotes in
    \enquote{quotes} can be
    troublesome.}

  He claimed:
  \foreignquote{german}{Quotes in
    \foreignquote{german}{quotes}
    can be troublesome.}
\end{FlushLeft}
```

He claimed: “Quotes in ‘quotes’ can be troublesome.”

He claimed: ≪Quotes in ‹quotes› can be troublesome.≫

7.2 Useful Symbols

There are a couple of mathematical symbols that cannot be found easily in LaTeX-symbol collections. In the following, a few such symbols are listed and a possible way is proposed of how to define them.

7.2.1 Markov Chains

One of the existing customary ways to describe that three random variables form a Markov chain is

```
\begin{IEEEeqnarray*}{c}
  X \markov Y \markov Z
\end{IEEEeqnarray*}
```

$$X \multimap Y \multimap Z$$

Here, the symbol "⊸" is defined as a combination of `\multimap` (⊸) and two minus-signs (−):

```
\newcommand{\markov}{\mathrel{\multimap}\joinrel\mathrel{-}%
                     \joinrel\mathrel{\mkern-6mu}\joinrel\mathrel{-}}
```

For this definition to work, beside `amsmath` also the package `amssymb` needs to be loaded.

7.2.2 Independence

To describe that two random variables are statistically independent, we propose the following symbol:

```
\begin{IEEEeqnarray*}{c}
  X \indep Y
\end{IEEEeqnarray*}
```

$$X \perp\!\!\!\perp Y$$

Accordingly,

```
\begin{IEEEeqnarray*}{c}
  X \dep Y
\end{IEEEeqnarray*}
```

$$X \not\perp\!\!\!\perp Y$$

denotes that X and Y are statistically dependent.

These two symbols are created from two `\bot` (⊥) signs:

```
\newcommand{\indep}{\mathrel{\bot}\joinrel\mathrel{\mkern-5mu}%
                    \joinrel\mathrel{\bot}}
\newcommand{\dep}{\centernot\indep}
```

For this definition to work, beside `amsmath` also the package `centernot` needs to be loaded.

7.2.3 Integration-d

The d in an integral is not a variable, but rather an operator. It therefore should not be typeset italic d, but Roman d. Moreover, there should be a small spacing before the operator:

```
\begin{IEEEeqnarray*}{c}
  \int_a^b f(x) \dd x = \int_a^b
  \ln\left(\frac{x}{2}\right)
  \dd x
\end{IEEEeqnarray*}
```

$$\int_a^b f(x)\,\mathrm{d}x = \int_a^b \ln\Big(\frac{x}{2}\Big)\,\mathrm{d}x$$

To make sure that this spacing always works out correctly, we recommend the following definition:

```
\newcommand{\dd}{\mathop{}\!\mathrm{d}}
```

7.2.4 Conditioning Bar

In a probability expression, the spacing around the conditioning bar is problematic:

```
\begin{IEEEeqnarray*}{l}
  \bigl[X^2=4 \big| Y=y\bigr]\\
  \bigl[X^2=4 \bigm| Y=y\bigr]
\end{IEEEeqnarray*}
```

$$\big[X^2=4\big|Y=y\big]$$
$$\big[X^2=4 \,\big|\, Y=y\big]$$

Note how `\big|` results in too little space, while `\bigm` adds too much space. We recommend the definition of the following commands:

```
\newcommand{\midk}[1]{\mspace{2mu} #1 \mspace{2mu}}
\newcommand{\middlek}[1]{\mspace{2mu} \middle#1 \mspace{2mu}}
\newcommand{\bigk}[1]{\mspace{2mu} \big#1 \mspace{2mu}}
\newcommand{\Bigk}[1]{\mspace{2mu} \Big#1 \mspace{2mu}}
\newcommand{\biggk}[1]{\mspace{2mu} \bigg#1 \mspace{2mu}}
\newcommand{\Biggk}[1]{\mspace{2mu} \Bigg#1 \mspace{2mu}}
```

These can be used in corresponding definitions of the conditional probability and more. For example,

```
\newcommand{\Prob}{\operatorname{\mathsf{P}}}
\newcommand{\Prvcond}[2]{\Prob\left[      #1 \middlek{|} #2 \right]}
\newcommand{\ePrvcond}[2]{\Prob[          #1 \midk{|}    #2 ]}
\newcommand{\bigPrvcond}[2]{\Prob\bigl[   #1 \bigk{|}    #2 \bigr]}
\newcommand{\BigPrvcond}[2]{\Prob\Bigl[   #1 \Bigk{|}    #2 \Bigr]}
\newcommand{\biggPrvcond}[2]{\Prob\biggl[ #1 \biggk{|}   #2 \biggr]}
\newcommand{\BiggPrvcond}[2]{\Prob\Biggl[ #1 \Biggk{|}   #2 \Biggr]}
```

defines a whole family of conditional probability commands with various bracket sizes that are either set automatically or manually:

```
\begin{IEEEeqnarray*}{l}
  \bigPrvcond{X^2=4}{Y=y}\\
  \Prvcond{\sum_{k=1}^n X_k^2
    = \alpha}{Y=y}
\end{IEEEeqnarray*}
```

$$\mathsf{P}\big[X^2=4\big|Y=y\big]$$
$$\mathsf{P}\left[\sum_{k=1}^{n} X_k^2=\alpha\middle|Y=y\right]$$

Another example is as follows:

```
\newcommand{\opD}{\operatorname{\mathsf{D}}}
\newcommand{\relDf}[2]{\opD\left( #1 \middlek{\|} #2 \right)}
```

resulting in:

```
\begin{IEEEeqnarray*}{c}
  \relDf{f_X^{(1)}}{f_X^{(2)}}
\end{IEEEeqnarray*}
```

$$\mathsf{D}\Big(f_X^{(1)}\Big\|f_X^{(2)}\Big)$$

Chapter 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. 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

moser@isi.ee.ethz.ch

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. Another always very useful source of information is *The Not So Short Introduction to LaTeX* by Tobias Oetiker.

Finally, I gratefully acknowledge the comments from (in alphabetical order) Helmut Bölcskei, Amos Lapidoth, Edward Ross, Omar Scaglione, and Sergio Verdú.

Stefan M. Moser

Index