The PracTEX Journal, 2012, No. 1
Article revision 2012/02/16

Avoid eqnarray!

Lars Madsen

Email daleif@imf.au.dk
Address Department of Mathematics
Aarhus University
Denmark

Abstract Whenever the eqnarray environment appears in a question or example of a problem on comp.text.tex, tex.stackexchange.com or other fora there is a high probability that someone will tell the poster not to use eqnarray. This article will provide some examples of why many of us consider eqnarray to be harmful and why it should not be used.

Introduction

When someone asks a question on comp.text.tex, tex.stackexchange.com or other fora about the eqnarray environment or shows an example using it, there will always be someone that instructs the poster to stop using eqnarray and use something better instead. This article provides an example-based overview of some of the reasons why many people consider eqnarray to be obsolete. Thus, this article can be used as a reference when a poster asks for an explanation.

The prerequisites for this article are a basic knowledge of LaTeX and knowledge of the syntax used by eqnarray. Experience with the environments from the amsmath package is a plus but not mandatory.

1 The basics

In plain LaTeX, the eqnarray environment is basically the only construction available for numbered multi-line equations. The eqnarray environment is similar to

Originally published in *The PracTEX Journal* 2006-4.

```
\begin{array}{rcl}
  ...
\end{array}
```

with the difference being that the first and last cell in each row are automatically typeset as display style mathematics, and not as text style math as it would be in the `array` environment; also, `eqnarray` supports automatic equation numbers.

The principal `eqnarray` usage is similar to this example:

```
\begin{eqnarray}
  y &=& (x+1)^2 \\  &=& x^2+2x+1
\end{eqnarray}
```

which results in (without the box):

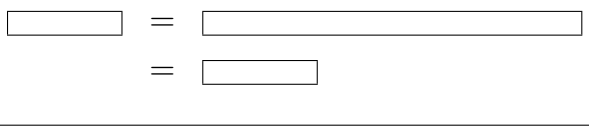
$$\begin{aligned} y &= (x+1)^2 && (1)\\ &= x^2+2x+1 && (2)\end{aligned}$$

In the examples that follow, we use the command `\dbx` instead of writing some meaningless arbitrary mathematical formula. `\dbx` is a simple macro, defined by the author, that writes a box to simulate some random mathematical material. Using an optional argument we can adjust the width of the box created by `\dbx`.

The reason for using simulated math instead of actually writing something is that removing the actual text makes the reader more aware of the actual problem, which is not the text but rather the construction/surroundings themselves. The example above will be shown like this instead:

```
\begin{eqnarray*}
  \dbx &=& \dbx[5cm] \\  &=& \dbx
\end{eqnarray*}
```

which results in:



2 Behold the problems

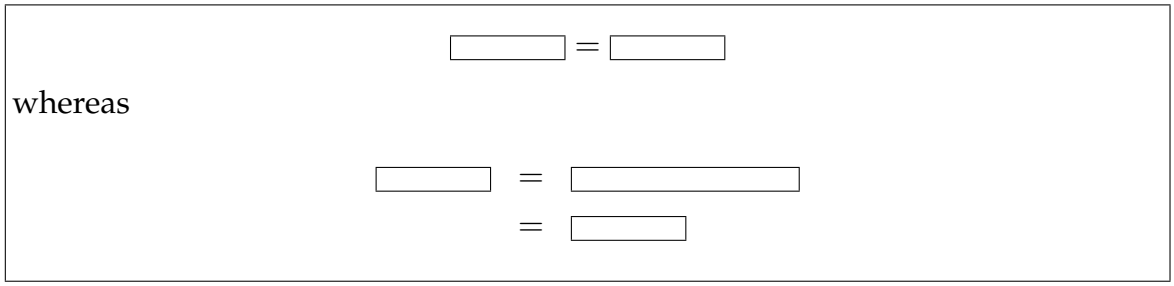
2.1 The primary problem: Spacing inconsistency

Most commonly, `eqnarray`-users write their displayed equations by mixing `eqnarray` and `eqnarray*` with `equation`, `\[...\]`, or `$$...$$` constructions. Some even mix it with environments from the `amsmath` package (though this is mostly seen when a document has been written by more than one author).

This mixing results in the primary problem with `eqnarray`—*spacing inconsistency*. In the following example we consider a single line equation versus a multi-line `eqnarray` equation.

```
\[ \dbx = \dbx \]
whereas
\begin{eqnarray*}
  \dbx &=& \dbx[3cm] \\  &=& \dbx
\end{eqnarray*}
```

which results in

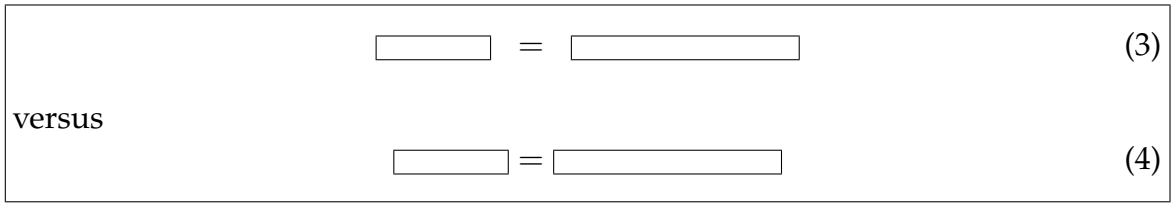


Can you spot the problem?

It is even more obvious when we place the same code using `eqnarray` and `equation` next to each other:

```
\begin{eqnarray} \dbx &=& \dbx[3cm]
\end{eqnarray}
  versus
\begin{equation} \dbx = \dbx[3cm]
\end{equation}
```

which results in



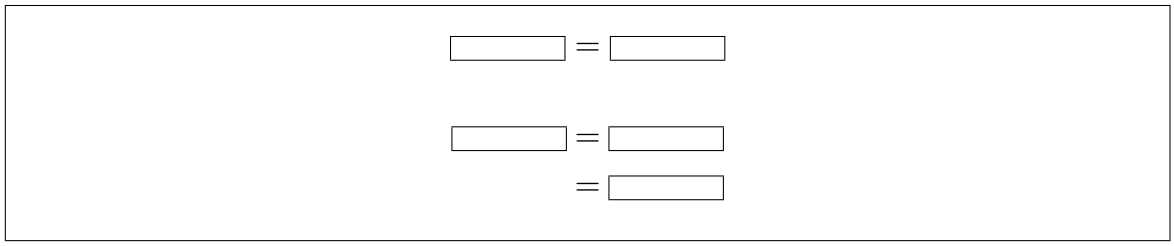
Can you see the difference?

We notice how the spacings around the ='s are inconsistent, i.e., not equal. Consistency being one of the key values in any good document design, the spacing around the = signs should be equal on both sides (not counting stretch), no matter which construction is used.

Since `eqnarray` is (naively) built on top of the `array` environment we still have the `\arraycolsep` space between columns, which then affects the spacing around the ='s in our case. We could change the value of `\arraycolsep`:

```
\setlength\arraycolsep{1.4pt}% some length
\[ \dbx = \dbx \]
\begin{eqnarray*}
  \dbx & = & \dbx  \\  &= & \dbx
\end{eqnarray*}
```

Resulting in:



Changing the value of `\arraycolsep`, however, will also change the appearance of any other construction that might be using `array`, so this does not suffice; see the following example.

```
Before the change:
\begin{eqnarray*}
  A &=& \left(\begin{array}{cc}\dbx&\dbx\\
              \dbx&\dbx\end{array}\right)
```

```
\end{eqnarray*}
after the change:
\setlength\arraycolsep{1.4pt}% some length
\begin{eqnarray*}
  A &=& \left(\begin{array}{cc}\dbx&\dbx\\
              \dbx&\dbx\end{array}\right)
\end{eqnarray*}
```

Resulting in:

Before the change:

$$A = \left(\begin{array}{cc}\dbx&\dbx\\ \dbx&\dbx\end{array}\right)$$

after the change:

$$A = \left(\begin{array}{cc}\dbx&\dbx\\ \dbx&\dbx\end{array}\right)$$

Some people argue that this larger spacing is a good thing, that it helps understanding the equations in question. For that to be true the author should do this with every single equation, whether the equation was written using `eqnarray` or not. Consistency above all. We can plainly see that `eqnarray` does not follow the spacing conventions Knuth set out in TEX, whereas both `equation` and `\[...\]` do.

Here is another example from a set of notes I have been editing (actual code from the original unedited notes).

```
\begin{eqnarray*}
 {\cal C}_{0} &\subseteq& {\cal C}\subseteq
 \sigma ({\cal  C}_{0},{\cal N}) ,
\end{eqnarray*}
```

$$\mathcal{C}_0 \subseteq \mathcal{C} \subseteq \sigma(\mathcal{C}_0, \mathcal{N}),$$

Which makes one wonder if LaTeX authors even notice the difference in spacing, or do they just accept it as a fact of life?

Even though `eqnarray` might not be recommended for one-liners, they do still appear quite a lot in the 'wild'.

As `eqnarray` is the only multi-line construction for plain LaTeX, what should be used instead? Short answer: Use the environments from the `amsmath` package, in particular the `align` environment.

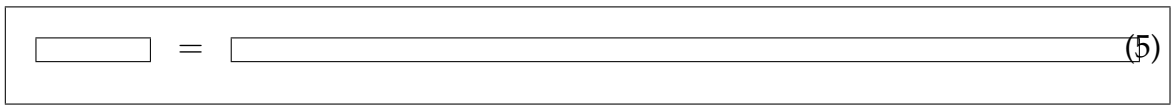
Longer answer: There are a few packages that can help including `nath`, `mathenv` and `amsmath`. Using `amsmath` is highly recommended since it is already included as part of every LaTeX installation.

For those not familiar with the `amsmath` package we present a few useful constructions in Appendix A.

2.2 Problem #2: eqnarray can overwrite equation numbers

Given a long formula which happens to fit within one line were it not for the equation number, `eqnarray` will happily just ignore the equation number, without any warnings.

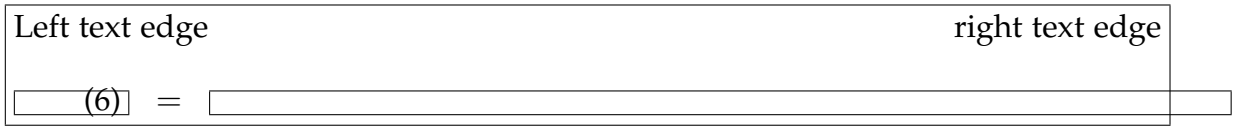
```
\begin{eqnarray}
  \dbx &=& \dbx[12cm]
\end{eqnarray}
```



It can get even worse. Assume we are using the `leqno` class option, i.e. equation numbers on the left. Then assume we have a math line that is slightly longer than the text width:[1]

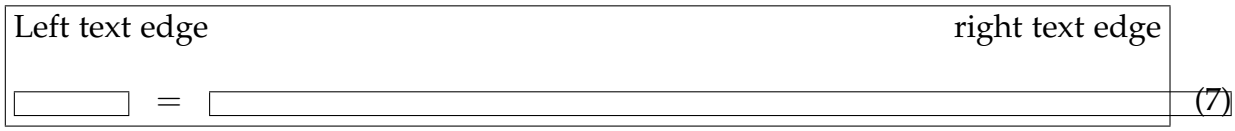
```
Left text edge \hfill right text edge%
\begin{eqnarray}
  \dbx &=& \dbx[13.5cm]
\end{eqnarray}
```

1. Example provided by Barbara Beeton.



No offence, but why on earth is `eqnarray` moving the equation number? Let us see what happens if we take the same example and switch back to equation numbers on the right:

```
Left text edge \hfill right text edge%
\begin{eqnarray}
  \dbx &=& \dbx[13.5cm]
\end{eqnarray}
```

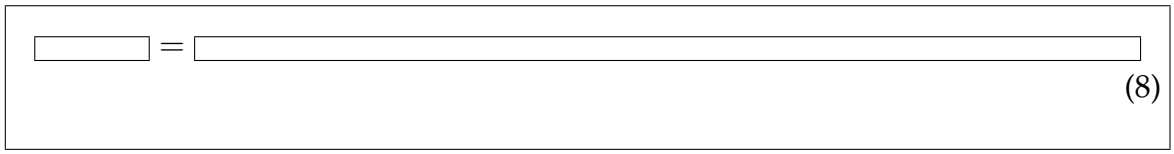


Sigh...

Well, at least that will teach authors to remember to break their equations properly.

At least the environments from the `amsmath` bundle take the equation number into consideration. Here is an example using `align`:

```
\begin{align}
  \dbx &= \dbx[12.5cm]
\end{align}
```



2.3 Problem #3: Silence of the labels

Part of my job is to process a preprint series published by my department. This brings me into contact with many different styles of LaTeX writing and usage. One thing that I frequently do (as part of my visual improvement procedures)

is convert `eqnarray` environments into `align` environments (or similar). This is where one starts to find the hidden label errors. Most often these occur when two or more people have been writing/editing the same file.

Here is the first example:

```
\begin{eqnarray}
  \dbx & = & \dbx \\
  \dbx & = & \dbx \label{eq:2} \nonumber
\end{eqnarray}
From equation (\ref{eq:2}) we conclude
\begin{equation}
  \dbx=42.
\end{equation}
```

So the author had an equation which he or she no longer wanted to have numbered (`\nonumber`). Which is perfectly reasonable, but the author neglected to check whether the now-dead label (`eq:2`) was referred to. The result is as follows:

$$\square = \square \tag{9}$$
$$\square = \square$$

From equation (10) we conclude

$$\square = 42. \tag{10}$$

Huh? This might end up as an interesting form of argumentation. It seems as if `eqnarray` actually steps up the equation counter at the start of every line (hence `\label` catches something) and when it encounters `\nonumber` it does not write any equation number and steps the equation counter one down again. On a side note, `equation` has the same problem if one mixes it with `\nonumber` (something which is *not* fixed by using `amsmath`).

The worst thing here is that `eqnarray` does this silently, without warnings, so if you do not know that this might happen you will never notice it unless someone carefully reads the article.

As it happens, I recently received an article which showed exactly the same problem in `eqnarray*`. Here one only has to place a label inside a non-numbering

eqnarray* (we use \theequation to show the current value of the equation number):

```
Equation number before the equation: \theequation
\begin{eqnarray*}
  \label{eq:4}
  \text{Inside the equation \theequation} & = & \dbx
\end{eqnarray*}
The reference is \eqref{eq:4}.
Equation number after the equation: \theequation
```

Resulting in:

Equation number before the equation: 10

$$\text{Inside the equation } 11 \quad = \quad \square$$

The reference is (11). Equation number after the equation: 10

Who smells a rat? So, even in eqnarray* the equation counter is stepped up, and later stepped down at the end of each line. As we have seen, this is a problematic approach.

2.4 Problem #4: The amsthm package vs. the eqnarray environment

If one uses the amsthm package, and its proof environment, then you will get automatic placement of an "*end of proof*" marker. Sometimes one ends a proof with a displayed formula and may want to place the end marker near the equation number. This may be achieved by simply issuing \qedhere on the last line of the formula.

```
\begin{proof}
  \dots
  \begin{equation*}
    a=0. \qedhere
  \end{equation*}
\end{proof}
```

Proof. ...

$$a = 0.$$ □

This handy little feature, as one might guess by now, does *not* work with `eqnarray`!

3 Solution

The best solution is to *not* use the `eqnarray` environment at all. Use the environments from `amsmath` instead. If in some case that will not do, the `mathenv` package reimplements `eqnarray` to work more rationally. It also removes the restraint on the number of columns in an `eqnarray`. (Unfortunately, `mathenv` is not compatible with certain useful modern packages, notably `siunitx`.)

Sadly we see many journals and publishers who still recommend (or at least mention) the use of `eqnarray` in their guides for authors.

A The `amsmath` package

For more information about `amsmath` see [2], [1] and [4] (in order of recommended reading). This appendix gives a few interesting constructions, mainly showing replacements for common `eqnarray` usage.

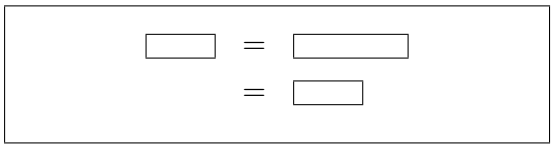
All of the following examples require `amsmath`, hence the document preamble must include:

```
\usepackage{amsmath}
```

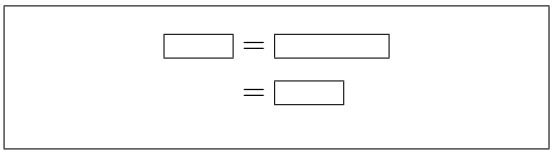
One thing to note about `amsmath` is that *every* environment from `amsmath` that provides equation numbers also has a `*`-version which does not. The package also includes an `equation*` environment which is missing from plain LaTeX.

Now the first thing we need is a replacement for `eqnarray`. We choose `align`, which has a slightly different syntax than `eqnarray`:

```
\begin{eqnarray*}
\dbx &=& \dbx[1.5cm]\\
     &=& \dbx
\end{eqnarray*}
```



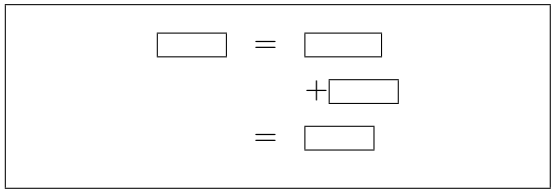
```
\begin{align*}
\dbx &= \dbx[1.5cm]\\
     &= \dbx
\end{align*}
```



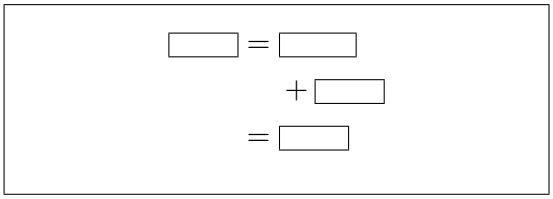
Note the reduced number of `&`'s.

Here is another common `eqnarray` construction and its `align` counterpart:

```
\begin{eqnarray*}
\dbx &=& \dbx[1cm]\\
     & & + \dbx   \\
     &=& \dbx
\end{eqnarray*}
```



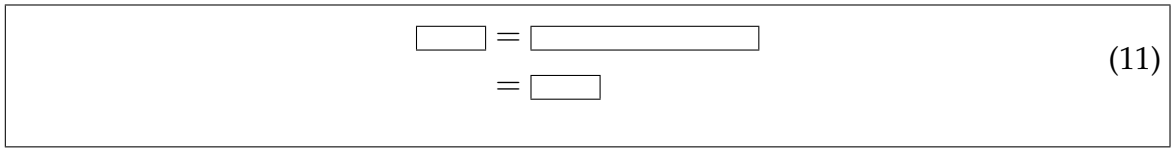
```
\begin{align*}
\dbx = {} & \dbx[1cm]\\
          & + \dbx   \\
     = {} & \dbx
\end{align*}
```



Notice the use of `{}` when the `&` is placed to the *right* of a relational symbol. Also note that the spacing around the + is correct in the `align` case but not when using `eqnarray`.

One construction not easily achieved with base LaTeX is a formula spread over several lines but with only one equation number for the entire formula. Again, this is easy using constructions from the `amsmath` package:

```
\begin{equation}
  \begin{split}
    \dbx & =\dbx[3cm] \\
         & =\dbx
  \end{split}
\end{equation}
```



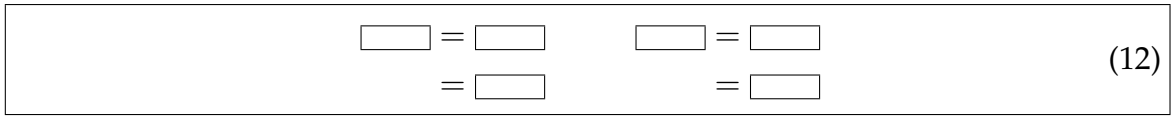
(11)

Notice how the equation number is vertically centred. The syntax for `split` is otherwise more or less the same as for `align*`.

The `amsmath` package also provides the `aligned` (and `alignedat`) environment, which is basically the full `align` environment, but for *inner* use. (Like `eqnarray`, `split` can only have one so-called alignment column, while `align` and `aligned` can have several.)

```
\begin{equation}
  \begin{aligned}

    \dbx & =\dbx &\qquad  \dbx & =\dbx \\
         & =\dbx &              & =\dbx
  \end{aligned}
\end{equation}
```

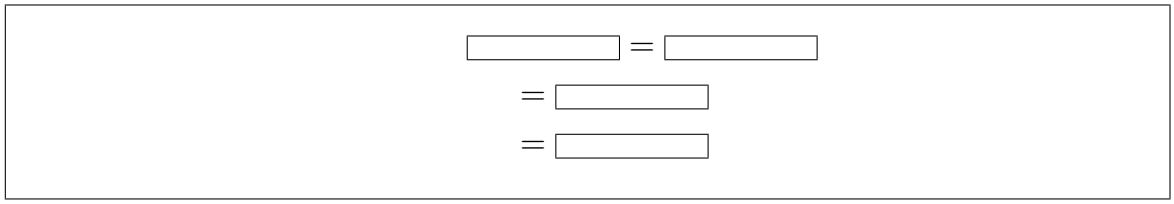


(12)

A.1 What about `\lefteqn`?

`amsmath` has no direct equivalent to `\lefteqn`, but the idea is still useful. To recap, using the `\lefteqn` macro inside `eqnarray`, one can force that particular line to be moved to the left:

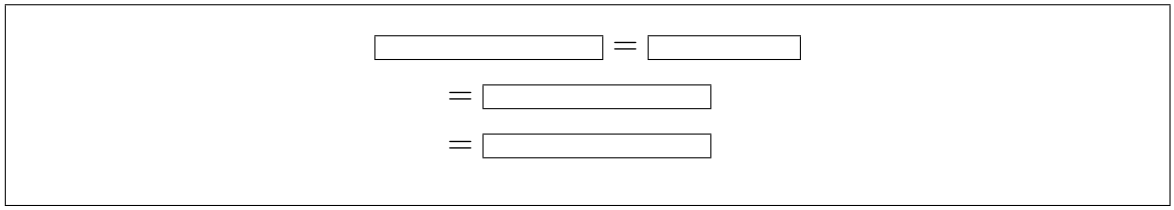
```
\begin{eqnarray*}
  \lefteqn{\dbx[2cm] = \dbx[2cm]} \\
  && = \dbx[2cm] \\
  && = \dbx[2cm]
\end{eqnarray*}
```



One usually uses this to mark the first line, and then give the impression of the rest of the lines being indented.

The `mathtools` package does provide an alternative, namely `\MoveEqLeft`:

```
\begin{align*}
  \MoveEqLeft \dbx[3cm] = \dbx[2cm] \\
       & = \dbx[3cm] \\
       & = \dbx[3cm]
\end{align*}
```



The idea is that the `\MoveEqLeft` marks an alignment point (which is what the ampersands follow), and then pulls the line backwards in a suitable fashion. It does *not* take any required arguments, unlike `\lefteqn`.

Acknowledgements

Special thanks to Barbara Beeton from the AMS for comments and suggestions for this revised version. Also many thanks to the various people who provided examples for the original version of the article.

References

[1] American Mathematical Society, *User's Guide for the* `amsmath` *Package*, 2002. Normally included in every LaTeX installation as `amsldoc`; also available via `http://mirror.ctan.org/macros/latex/required/amslatex/math`.

[2] Michael Downes, *Short Math Guide*, 2002. Short introduction to the `amsmath` and `amssymb` packages. `ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf`

[3] Morten Høgholm, Lars Madsen, Will Robertson and Joseph Wright (maintainers), *The mathtools package*, 2011. Various extensions to `amsmath` and others. `http://mirror.ctan.org/macros/latex/contrib/mh`.

[4] Herbert Voß, *Math mode*, 2006. Extensive summary describing various mathematical constructions, both with and without the `amsmath` package. `http://mirror.ctan.org/info/math/voss/mathmode/Mathmode.pdf`.