III Maths in LaTeX: Part 1, Back to Basics

R. A. Bailey
Goldsmiths' College, University of London

1 Introduction

The book *LaTeX: A Document Preparation System* by Leslie Lamport is rather coy about Mathematics. It simply does not reveal the full range of Mathematical expressions that can be correctly typeset without going outside LaTeX. The result is that some Mathematical authors, while attracted to the generic mark-up of LaTeX, believe that they need to use plain TeX or AMSTeX to write their documents.

This sequence of tutorials seeks to correct that impression, by explaining what Mathematical expressions can be typeset with LaTeX without the need for the `amstex` package. Perhaps this will provoke someone else to write a tutorial on that package. The first part is mostly, but not entirely, devoted to things which you can find in *The Manual*, even though you may have overlooked some of them. Succeeding parts (in the next and later issues of *Baskerville*) will be mostly about Mathematical goodies provided by TeX but upon which *The Manual* is silent, even though they are necessary and quite easy to use. The final part will deal with arrays, concentrating on their use in Mathematics.

These are tutorials, so I expect you, the reader, to do some work. Every so often comes a group of exercises, which you are supposed to do. Use LaTeX to typeset everything in the exercise except sentences in italics, which are instructions. If you are not satisfied that you can do the exercise, then write to me with hard copy of your input and output (no email address before we go to press, I'm afraid): I will include a solution in the following issue of *Baskerville*.

A word on fonts. Fonts in Mathematics are handled differently in LaTeX 2.09, in NFSS, and in LaTeX 2ε. Rather than compare these systems every time that I mention fonts, I shall limit myself to LaTeX 2.09. With any luck, this will enrage some knowledgeable person enough to write an article on handling of Maths fonts in different flavours of LaTeX.

2 What does it look like?

2.1 Maths Mode

(LA)TeX has a special state, called *Maths mode*, which it must be in to recognize Mathematical expressions and typeset them properly. Maths mode in LaTeX is everything between `\(` and `\)`, or, alternatively, everything between `$` and `$`. The parentheses are better for trapping errors, because it is obvious whether the left or right one is missing, if any. A missing `$` causes (LA)TeX to swap Maths mode and ordinary mode from then onwards, giving strange output but no errors until it eventually meets something like `x^2` that it cannot interpret in the wrong mode. On the other hand, the dollar signs are easier to type, and easier to see in your input file.

In Maths mode most symbols are typeset as if they represent single-letter variables. A string of three letters will be set as if those three variables should be multiplied together. Fancy features like kerns and ligatures, which are used in normal text to help the reader interpret letter-strings as words, are turned off. Letters are set in the special font known as *Maths italic* which is usually used for variables.

Almost all spaces that you type are ignored. (LA)TeX thinks that it knows better than you do how Mathematics should be spaced, and it is probably right to think so.

Don't stay in Maths mode for too long just because you are too lazy to type a few `$` signs. Everything between the `$`s should be Maths. A common mistake by beginners is to forget that a punctuation sign, like a comma, may have a different meaning in Maths from its meaning in text. In

the scalars a, b and c

we have a textual list containing three mathematical objects, so the input file contains

```
the scalars $a$, $b$ and $c$
```

That comma is a textual one. The lazy typist types

```
the scalars $a, b$ and $c$
```

and obtains

the scalars a, b and c

On the other hand, in

the vector (a, b, c)

there is a single Mathematical object, so it is correct to type

```
the vector $(a,b,c)$
```

or, equally well,

```
the vector $(a, b, c)$
```

These commas are part of the Mathematical notation.

2.2 Basic symbols

The basic symbols are the numerals `1`, `2`, ..., the Latin letters `a`, `b`, ..., `z`, `A`, ..., `Z`, and the Greek letters `\alpha`, `\beta`, `\gamma`, ..., `\omega`, `A`, `B`, `\Gamma`, ..., `\Omega`. If you don't know the standard English spellings of Greek letters, look on page 43 of *The Manual*. Upper-case Greek letters which are conventionally the same as their Latin equivalents do not have special commands. Some Greek letters have variants: `\varepsilon`, for example.

The obvious symbols for operators are the keyboard symbols `+` and `-`. If you forget to go into Maths mode (a common temptation when typing a table of data), the symbol `-` will not look like a minus sign. Outside Maths mode the `+` will look like a plus sign, but the spacing will be wrong. In Maths mode (LA)TeX knows what is the proper spacing to put around binary operators like `+` and `-`; it also knows the proper spacing to surround binary relations like `=`. Try typing the following both inside Maths mode and outside it, and compare the results.

```
1 +2 =  3          4-1 = 3
1 -4 = -3         -2+7 =+5
```

Also try `>` outside Maths mode: you may be surprised.

2.3 Sub- and superscripts

Subscripts are introduced with `_`: for example, `x_n` gives x_n. If there is more than one thing in the subscript you have to use braces, as in `x_{n+1}` for x_{n+1}. You can type `x_{n}` for x_n if you want, but it makes your input file less readable.

Superscripts are done similarly, using `^`: thus `y^3` for y^3 and `y^{-1}` for y^{-1}.

A sub- and superscript can be put on the same symbol in either order: `x_n^2` and `x^2_n` both produce x_n^2. Double subscripts or superscripts are obtained by using braces in the obvious way: `x_{n_2}` and `n^{m^2}`.

To put a sub- or superscript *before* a symbol, precede it with `{}`. Otherwise the sub- or superscript attaches itself to the previous thing, which may well be something like $+$ or $=$.

In an expression such as $(X+Y)^2$, strictly speaking TeX thinks it is putting the superscript on the right parenthesis if you type `(X+Y)^2`, and it positions the superscript in accordance with that thought. If this really offends you, you can force TeX to share your logic by typing `{(X+Y)}^2`, but you may not always prefer the result.

2.4 Modifying symbols

To turn x into x' type `x'`. You do not need to think of the prime as a superscript.

Some common modifiers are exemplified in

`\bar{x}`	$\bar{x}$	`\tilde{x}`	$\tilde{x}$
`\hat{x}`	$\hat{x}$	`\vec{x}`	$\vec{x}$

A few more such decorations are shown on page 51 of *The Manual*. If any of them is used over an i or a j then the dotless versions of those letters should be used: `\imath` and `\jmath`.

There are wide versions of `\hat` and `\tilde`:

`\widehat{a+b}`	$\widehat{a+b}$
`\widetilde{1-\theta}`	$\widetilde{1-\theta}$

There are also wide versions of `\bar` and `\vec` but with less obvious names: I'll cover these in a later tutorial.

Logically, a decoration such as `\hat` may modify the whole of a subscripted expression such as x_2; you usually mean 'the estimate of x_2' rather than 'the second part of $\hat{x}$'. However, both $\hat{x_2}$ and $\widehat{x_2}$ simply look wrong, so you have to let aesthetics triumph over logic and type `\hat{x}_2`.

2.5 Dots

To get a line of dots to show that some items have been missed out, use `\ldots` if the missing items are normally aligned on their baselines, such as letters, or `\cdots` if the missing items are normally aligned on the centreline, such as binary operators. If the missing items are part of a textual list, don't forget to come out of Maths mode and to put a comma at the end of the dots.

```
for $i=1$, $2$, \ldots, $10$
the vector $(x_1, x_2, \ldots, x_n)$
$a_1 + a_2 + \cdots + a_n$
$y_1 = y_2 = \cdots = y_7$
```

If you think that the dollar signs round the numerals in the first example are unnecessary, try embedding that phrase in a piece of italic text.

2.6 Square roots

Type `\sqrt{2}` to obtain $\sqrt{2}$. The same technique works for more complicated expressions than 2: you don't have to do anything to make the root sign the right size. For example,

`\sqrt{n^2+6}` $\sqrt{n^2+6}$

Other roots, such as cube roots, are obtained by putting in an optional argument:

`\sqrt[3]{8} = 2` $\sqrt[3]{8}=2$

The simple symbol for a square root is `\surd`.

Don't abuse TeX's wizardry by using `\sqrt` for a large expression in text or in a complicated display. The mess obscures the message.

2.7 Displayed Maths

To get a single line of displayed Maths, type the contents between `\[` and `\]`. You should not start a paragraph with displayed Maths, but may end one. If the displayed Maths is in the middle of a paragraph, remember not to leave blank lines around it in your input file.

Displayed Maths may also be typed between `$$` and `$$`, but the effect is not quite the same. For example, the document option `fleqn` aligns displayed Maths on the left if you use `\[` and `\]`, but not if you use `$$`.

To put a short piece of text in displayed Maths, insert it in `\mbox`, remembering to include any necessary spaces that would be ignored in Maths mode.

```
\[ a=b \mbox{ if } c=d \]
```

Don't try to use `\mbox` in a similar way to put short text between pieces of Maths in text: it inhibits line-breaks.

2.8 Words as labels

Sometimes you want to attach natural-language words to Mathematical symbols to label them. For example, you might have analogous quantities associated with the rows and columns of a rectangular array, and wish to indicate this by using the same symbol, say Q, with different subscripts. It simply will not do to type `Q_{rows}`, because this gives Q_{rows}, where the subscript looks like the product of r by o by .... And it is no good putting `rows` in an `\mbox`, because it will come out too big. Once something has been put in a box, it doesn't change size. You have to type `Q_{\rm rows}` to get $Q_{\rm rows}$. (Did you remember the caveat about fonts?)

If this seems too much trouble, you might decide to abbreviate to Q_r and Q_c. But this will not do either, because the subscripts look like variables into which numbers, say, could be substituted. If you don't want to mislead your readers, you should type `Q_{\rm r}`.

2.9 Fractions

A built-up fraction is made with `\frac`:

`\frac{n}{m}` $\frac{n}{m}$

This comes out larger in displayed Maths than in text. It is better to use the solidus, as in n/m, for most fractions in text, with the exception of a few simple common fractions like $\frac{1}{2}$.

Of course, fractions can be put inside other fractions with no bother:

```
\frac{a(b+c)}{5 + \frac{1}{xy}}
```

$$\frac{a(b+c)}{5 + \frac{1}{xy}}$$

2.10 Binary operators

In the golden olden days of golf-ball typewriters, it was a luxury to a Mathematician to have the symbol for direct sum, or for union. (LA)TEX not only has the symbols; it knows that they are operators, and gives them the correct spacing for infix operators, and has reasonably good ideas about where to break lines near them. A few of the common ones are:

`+`	$+$	`-`	$-$	`\pm`	$\pm$
`\times`	$\times$	`\div`	$\div$	`\oplus`	$\oplus$
`\cup`	$\cup$	`\cap`	$\cap$	`\wedge`	$\wedge$.

There are many more on page 44.

In fact, (LA)TEX is even cleverer than this. If a binary operator doesn't find itself between two things it can operate on then it becomes a simple symbol, and spaces and line-breaks adjust accordingly. You should have noticed this if you did the exercise suggested above.

2.11 Binary relations

(LA)TEX also knows about infix relations, such as

`=`	$=$	`\in`	$\in$	`\subset`	$\subset$
`<`	$<$	`\leq`	$\leq$	`\perp`	$\perp$.

More are shown on page 44. Don't confuse $\in$ with either of the epsilons.

Compare `\mid` with `|`. The former is a relation, while the latter is just a symbol. So which should you use for 'divides'?

Relations can be negated by preceding them with `\not`:

`Z_2 \times Z_2 \not\cong Z_4` $\quad Z_2 \times Z_2 \not\cong Z_4$

This doesn't work quite right for $\in$, so there is the special command `\notin`. Also, `\ne` is a useful shorthand for `\not=`.

2.12 Fonts in Maths

(Did you remember the caveat about fonts?)

For something like script letters use `\cal`, as in `${\cal F}(x)$` for $\mathcal{F}(x)$. The braces give the scope of `\cal`: for a single Mathematical letter such as $\mathcal{H}$ you can get away with `$\cal H$`. Only upper-case Latin letters may be modified by `\cal`.

In some branches of Mathematics, constants are shown in Roman type. So the base of natural logarithms is `{\rm e}`.

For bold letters, you can use `\bf` to modify Latin letters and upper-case Greek ones:

`{\bf Mv} = a{\bf w}` $\quad \mathbf{Mv} = a\mathbf{w}$

For lower-case Greek letters, and for non-letters, you have to use a cumbersome construction:

`\mbox{\boldmath $\lambda$}` $\quad \boldsymbol{\lambda}$

Because of the box, this does not change size properly in sub- and superscripts.

2.13 Writing Maths

The ability to produce beautiful Mathematical formulae is no licence to produce poor Mathematical writing. Remember that relations are verbs. It is impossible to parse the sentence

> Therefore $n = 56$ is the sample size.

but

> The equation $x^2 + 9 = 0$ has no real roots.

is fine.

Don't start a sentence with notation: the reader doesn't get the right visual clue. If possible, avoid putting notation immediately after *any* punctuation, unless it is part of a list. This saves the reader from having to work out if the punctuation is Mathematical or textual. Similarly, avoid abbreviations like 'iid' and 'e.g.' which might be mistaken for notation at a first glance.

3 Exercises

Exercise 1 The zeros of the quadratic ax^2+bx+c are

$$\frac{-b\pm\sqrt{b^2-4ac}}{2a}.$$

Exercise 2 The upper 5% point of the χ^2_6 distribution is 12.592.

Exercise 3 If $\nu=n_1+n_2-2$ and

$$s^2=\frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{n_1+n_2-2}$$

then

$$\frac{\bar{X}_1-\bar{X}_2}{s\surd(\frac{1}{n_1}+\frac{1}{n_2})}$$

is distributed as t_ν.

Exercise 4 By choosing bases, it follows that the subspaces $Z_1, \ldots, Z_r$ span V; hence it follows that V is the direct sum $V=Z_1\oplus\cdots\oplus Z_r$, as asserted.

Exercise 5 If $\mathcal{M}$ and $\mathcal{N}$ are subspaces of a finite-dimensional inner product space $\mathcal{V}$ then

$$(\mathcal{M}+\mathcal{N})^\perp=\mathcal{M}^\perp\cap\mathcal{N}^\perp$$

and

$$(\mathcal{M}\cap\mathcal{N})^\perp=\mathcal{M}^\perp+\mathcal{N}^\perp.$$

Moreover, $\mathcal{M}^\perp\cong\mathcal{V}/\mathcal{N}$.

Exercise 6 The sum of squares for the linear model $V_{\text{protein}}+V_{\text{fishmeal}}$ is 1559378.

Exercise 7 The usual regression equation is $\mathbf{Y}=\mathbf{X}\beta+\varepsilon$, where $\mathbf{Y}$ is an $n\times 1$ vector, $\mathbf{X}$ is an $n\times p$ matrix, β is the $p\times 1$ vector of unknown parameters, and ε is the $n\times 1$ vector of random errors. The least-squares estimate $\widehat{\beta}$ of the parameters is given by

$$\widehat{\beta}=(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}.$$

Exercise 8 The T-orders are $p(x)^{e_1}$, $p(x)^{e_2}$ and $p(x)^{e_3}$, where $e_1>e_2\geq e_3$. This implies that $p(x)^{e_1}\mid\eta(x)^{e_1-d}$ and hence that $\eta(x)=\psi(x)p(x)^d$ for some polynomial $\psi(x)$.

Exercise 9 We have $t\in A\setminus B$ if and only if

$$t\in A \text{ and } t\notin B.$$

Exercise 10 Pascal's triangle is based on the identity

$${}^{n-1}\mathrm{C}_k+{}^{n-1}\mathrm{C}_{k-1}={}^{n}\mathrm{C}_k.$$

XI Maths in LaTeX: Part 2, Getting more Serious

R. A. Bailey
School of Mathematical Sciences
Queen Mary and Westfield College
Mile End Road
London E1 4NS

Recall

This is the second in a sequence of tutorials on typesetting Mathematics in LaTeX. It includes some things which can be found in *LaTeX: A Document Preparation System* by Leslie Lamport, but I am gradually working in more things which, while straightforward and necessary for Mathematical work, are not in *The Manual*. In case you missed the first tutorial, two warnings are now repeated.

I expect you, the reader, to do some work. Every so often comes a group of exercises, which you are supposed to do. Use LaTeX to typeset everything in the exercise except sentences in italics, which are instructions. If you are not satisfied that you can do the exercise, then tell me. Either write to me at Queen Mary and Westfield College (my full address is given at the end of this article) with hard copy of your input and output, or email me at `r.a.bailey@qmw.ac.uk` with a copy of the smallest possible piece of LaTeX input file that contains your attempt at the answer. In either case I will include a solution in the following issue of *Baskerville*: you will remain anonymous.

A word on the controversial issue of fonts. Fonts in Mathematics are handled differently in LaTeX 2.09, in NFSS, and in the new standard LaTeX, LaTeX 2_ε. Rather than compare these systems every time that I mention fonts, I limit myself to LaTeX 2.09. When you upgrade to LaTeX 2_ε, all these commands will still work, so long as you use the standard styles `article`, `report` and `book`.

2 What does it mean?

2.1 Arrows

Arrows are relations. Four of them are

`\leftarrow`	$\leftarrow$	`\longleftarrow`	$\longleftarrow$
`\Leftarrow`	$\Leftarrow$	`\Longleftarrow`	$\Longleftarrow$

and eight others can be obtained by replacing `left` by `right` (all versions) or by `up` or `down` (not the long ones). A full list is given on page 45 of *The Manual*.

Because of its frequent use in defining functions, `\rightarrow` has the short alternative name `\to`. It should not be confused with `\mapsto`, which is also used in defining functions.

`x\mapsto g^{-1}xg` $x \mapsto g^{-1}xg$

The relation `\iff`, which does indeed stand for the relation 'iff', is not quite the same as `\Longleftrightarrow`: it has a little more space on either side.

```
gh^{-1} \in K \iff g \in Kh
```

$$gh^{-1} \in K \iff g \in Kh$$

2.2 Fences

Mathematical typesetters use the word *fence* for anything like a bracket that comes with a mate to enclose part of a formula. TeX calls them *delimiters*. Obvious examples are parentheses, brackets and braces, produced with `(`, `)`, `[`, `]`, `\{` and `\}` respectively. Angle brackets are produced with `\langle` and `\rangle`:

`\langle u,v \rangle` $\langle u,v\rangle$

Don't try to use the keyboard symbols `<` and `>` for this: apart from the fact that they do not look right in typeset Maths, (LA)TeX does not think that they are fences and may well split the line between the $<$ and the u.

Use `\lfloor`, `\rfloor`, `\lceil` and `\rceil` to obtain 'floor' and 'ceiling':

`\lceil 3.75 \rceil = 4` $\lceil 3.75 \rceil = 4$

Other fences are shown on page 48 of *The Manual*.

Fences need to grow to enclose large formulas. They will automatically grow to the correct size if you preface the opening fence with `\left` and the closing fence with `\right`. The `\left`s and `\right`s must come in properly matching pairs, but the fences which they qualify need not obviously match. Compare

$\frac{1}{n} \in (0,1]$ with $\frac{1}{3} \in \left(0, \frac{1}{2}\right]$:

the second half-open interval is produced with

```
\left( 0, \frac{1}{2} \right]
```

The solidus `/` is really a binary operator, but it is treated by TeX as a fence, partly because traditional typesetters do not put the same amount of space around it as they do around other binary operators, partly because it needs to grow when it is between two tall formulas. But it can't grow unless it is matched with another fence, and it doesn't need one. So TeX makes the full stop into an invisible fence (called the *null delimiter*) when it is preceded by `\left` or `\right`.

```
a(b+c) \left / \frac{1}{xy} \right.
```

$$a(b+c) \left/ \frac{1}{xy} \right.$$

How would the above expression be different if you typed `\left. a(b+c) \right /`...?

Sometimes you need to use `\left` and `\right` just to tell TeX that you are using fences, even if you do not need them to change size. A good example is $|\ \ |$ used for the modulus or cardinality functions. If you type `|+3|`, TeX will typeset the first $|$ as if it is being added to the 3.

```
| +3 |
```
$$| + 3|$$

```
\left| +3 \right|
```
$$\left|+3\right|$$

2.3 Standard functions with English names

Some standard functions have written names with two or more letters, based on their full English name. An example is cos for 'cosine'. It is no good simply typing `cos`, for then the output will look like *c* multiplied by *o* multiplied by *s*. So there are standard commands such as `\cos`, `\sin`, `\log`, `\exp` and `\dim`.

`\cos\pi = -1` $\cos\pi = -1$

The following ten standard functions

```
\max  \sup  \limsup  \lim  \det
\min  \inf  \liminf  \gcd  \Pr
```

can have expressions above and/or below them to show what range of variables they apply to: these are typed in as if they were super- or subscripts.

```
\lim_{x \to 0} \frac{\sin x}{x} = 1
```

$$\lim_{x \to 0} \frac{\sin x}{x} = 1$$

`\max_{i=1}^{n} \theta_i` $\displaystyle\max_{i=1}^{n} \theta_i$

In displayed Maths the range expressions come above and below the name of the standard function; in text they come somewhat to the right: $\max_{i=1}^{n} \theta_i$.

The standard function 'modulo' has two forms:

`7 = 10 \bmod 3` $7 = 10 \bmod 3$

`7 = 10 \pmod 3` $7 = 10 \pmod 3$

The remaining standard functions are listed on page 46 of *The Manual*.

2.4 Large repeated binary operators

When a binary operator is commutative and associative it often has a a special symbol to show its repeated application: $\sum$ for repeated $+$, $\bigcap$ for repeated $\cap$, and so on. Some of these have special commands in TEX:

`\sum`	$\sum$	`\bigcap`	$\bigcap$	`\bigoplus`	$\bigoplus$
`\prod`	$\prod$	`\bigcup`	$\bigcup$	`\bigotimes`	$\bigotimes$

(Note that the operator `\sum` is *not* the same as the Greek letter `\Sigma`.) More are shown on page 45 of *The Manual*. Each of these has two sizes: the big one is used in displayed Maths (except inside arrays and fractions) and the small one is used in text. Each can also take sub- and superscripts to show the range of operation: as with the ten listed standard functions, these sub- and superscripts appear above and below the operator in displayed Maths (except inside arrays and fractions) and a little to the right otherwise.

```
x_1 + \cdots + x_n = \sum_{i=1}^{n} x_i
```

$$x_1 + \cdots + x_n = \sum_{i=1}^{n} x_i$$

```
M_1 \vee\cdots\vee M_t = \bigvee_1^t M_i
```

$$M_1 \vee\cdots\vee M_t = \bigvee_1^t M_i$$

But in text: $\alpha_1\alpha_2\ldots\alpha_m = \prod_{j=1}^m \alpha_j$.

2.5 Integrals

There are two integral signs:

`\int`	$\int$	`\oint`	$\oint$

These behave somewhat like the large repeated operators in that they have a larger size in displayed Maths and their limits are typed in as sub- and superscripts. However, these limits stay in the same position even in displayed Maths.

2.6 More symbols

There are some miscellaneous Mathematical symbols that we have not covered elsewhere. Important ones include:

`\emptyset`	$\emptyset$	`\forall`	$\forall$	`\Box`	$\Box$
`\infty`	∞	`\exists`	$\exists$	`\partial`	∂

More are shown on page 45 of *The Manual*.

Unfortunately, some of these, such as `\Box`, are not standard in the new standard LATEX. To continue using such symbols after you have upgraded, you must include the package `amsfonts`, if available, or the package `latexsym`. Some operators and relations have been similarly banished, and can be rescued in just the same way.

2.7 Punctuation

In Maths mode, (LA)TEX treats a full stop as an ordinary symbol, so that decimal points look right. However, the comma and semi-colon are treated as punctuation, and get a little space after them, but not so much as the space they usually get in text. To suppress the space, put the comma or semi-colon in a pair of braces.

The colon is treated as a relation, because of its frequent use in defining sets. To obtain a colon as a piece of punctuation, type `\colon`. Many people think that the punctuation form is more correct for defining functions.

2.8 Braces for grouping

Grouping has two extra properties in Maths mode. In the first place, it can prevent line-breaking. (LA)TEX is usually very good at knowing where to break a line within a Mathematical expression, but it doesn't always do it exactly to your liking. For example, in Exercise 7 in the first tutorial, one equation was broken after a $+$ sign, leaving the single character ε on the next line. This could have been prevented by enclosing the whole right-hand side of the equation in braces: TEX never breaks a line between grouping braces in Maths mode. Use this force sparingly: the more line breaks that you ban the harder is it for TEX to build beautiful paragraphs.

The second property is more subtle: injudiciously placed grouping braces can destroy TEX's fine knowledge of what is a relation, an operator etc., and thus prevent it from applying the proper spacing. `2{+3}` is not the same as `2+3` and neither is `2{+}3`; while `{}-4` is different from `-4`. Likewise, `{\cos} \theta` is not the same as `\cos \theta`: the former turns `\cos` from an operator into a symbol.

Some TeXies recommend always putting `{}` after a command with no argument. That advice can be disastrous in Maths mode. Look at

`\sum{}_{1}^{3} y_j` ${\sum}_1^3 y_j$

and compare

`k \geq{} -b` $k \geq{} -b$

`k \geq   -b` $k \geq -b$

In the first example the sub-and superscripts have been placed on the `{}`, not on the `\sum`: in particular, they do not move to the correct place in displayed Maths. In the second, the minus sign has become a binary operator between `{}` and b instead of a unary operator on b.

This apparently awkward property of braces can be turned to advantage when TeX's first interpretation is not the correct one, as we have already seen in some examples.

2.9 Ties

The tie `~`, which prevents line breaks both within and without Maths mode, can be used to make your Mathematical text easier to read. A piece of notation consisting of a single letter or symbol should almost always be tied to one of its neighbours. If the notation names a noun, tie it to the noun: `group~$G$` and `element~$h$`. If it is the subject or single direct object of a verb, tie it to the verb:

```
If $v$~annihilates~$W$ then ...
```

If it follows a preposition, tie it to the preposition: `of~$\Lambda$`, `in~$T$`, `onto~$Y$`. If an adjective qualifies it, tie it to the adjective: `positive~$\delta$`. A notational adjective, particularly a number, should be tied to its following noun: `$n$~points`, `$21$~lines`. Be careful about the beginnings and ends of lists:

```
Only one of $A$, $B$ and~$C$ occurs ...
For $i=1$, $2$, \ldots,~$n$, put ...
```

Finally, use a tie if a small piece of notation ends a sentence, to prevent a line beginning something like

B. Therefore ...

which makes it look like the second item in a list.

Try to get in the habit of inserting these ties when you first type the text: don't leave them until you see bad line-breaks.

Of course, like all rules in typography, some of the above rules conflict, so you have to use common sense about which ones to follow. Also, there will always be times when the paragraph simply cannot be broken into lines nicely if all the ties are observed: wait until the final run and then relax the least important ones.

Exercises

Exercise 11 The function $f\colon Z \to Z$ defined by $f\colon n \mapsto n^2$ is neither injective nor surjective. However, the function $g\colon R^+ \to R^+$ given by $g(x) = \sqrt[7]{x}$ is both.

Exercise 12 If $\{x_1,\dots,x_n\}$ and $\{y_1,\dots,y_m\}$ are bases for $\mathcal{X}$ and $\mathcal{Y}$ respectively then

$$\left\{x_i \otimes y_j : i = 1, \ldots, n,\ j = 1, \ldots, m\right\}$$

is a basis for $\mathcal{X} \otimes \mathcal{Y}$.

Exercise 13 If $A = \{n \in N : n \text{ is prime}\}$ and if $B = \{n \in N : n \text{ is even}\}$ then $|A \cap B| = 1$.

Exercise 14 If z is any real number then $|+z| = |-z|$.

Exercise 15 *Express the following use large binary operators instead of dots. Try them both in text and in display.*

$x_1 + \cdots + x_{153}$	$1 + 2 + \cdots + r$
$1 + 1/2 + 1/4 + \cdots$	$p \times (p-1) \times \cdots \times 1$
$\mathcal{U}_1 \otimes \cdots \otimes \mathcal{U}_m$	$T_4 \cup T_6 \cup \cdots \cup T_{10}$
the sum of γ for $\gamma \in \Gamma$	$P_1 \wedge \cdots \wedge P_r$

Exercise 16

$$\prod_{k\geq 0}\frac{1}{(1-q^k z)} = \sum_{n\geq 0} z^n \bigg/ \prod_{1\leq k\leq n}(1-q^k)$$

Exercise 17 *Redo Exercise 3 from the first tutorial without using built-up fractions. You may want to change the way the square root is shown.*

Exercise 18

$$\sum_{n=1}^{N} n^3 = \left(\frac{N(N+1)}{2}\right)^2$$

Exercise 19 $\int_1^2 \frac{1}{x} dx = [\log x]_1^2 = \log 2$.

Exercise 20 $\lim_{n\to\infty}(1+\frac{x}{n})^n = \exp(x)$.

Exercise 21 If Z is a normal random variable with mean μ and variance σ^2 then

$$\Pr(Z < x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right).$$

Exercise 22

$$\frac{\partial \mathrm{e}^{-xt}}{\partial t} = -x\mathrm{e}^{-xt}.$$

VII Maths in LaTeX: Part 3, Different Sorts of Mathematical Object

R. A. Bailey
Queen Mary and Westfield College, University of London

1 Recall

This is the third in a sequence of tutorials on typesetting Mathematics in LaTeX. The first two appeared in issues 4.4 and 4.5 of *Baskerville*. The series includes some things which can be found in [[5]], but I am working in more things which, while straightforward and necessary for Mathematical work, are not in [[5]] or [[6]]. In case you missed the first two tutorials, two warnings are now repeated.

I expect you, the reader, to do some work. Every so often comes a group of exercises, which you are supposed to do. Use LaTeX to typeset everything in the exercise except sentences in italics, which are instructions. If you are not satisfied that you can do the exercise, then tell me. Either write to me at

School of Mathematical Sciences
Queen Mary and Westfield College
Mile End Road
London E1 4NS

with hard copy of your input and output, or email me at `r.a.bailey@qmw.ac.uk` with a copy of the smallest possible piece of LaTeX input file that contains your attempt at the answer. In either case I will include a solution in the following issue of *Baskerville*: you will remain anonymous if you wish.

A word on the controversial issue of fonts. Fonts in Mathematics are handled differently in LaTeX 2.09, in NFSS, and in the new standard LaTeX, LaTeX 2ε. Rather than compare these systems every time that I mention fonts, I usually limit myself to LaTeX 2.09. When you upgrade to LaTeX 2ε, all these commands will still work, so long as you use the standard styles `article`, `report` and `book`. In the 'Answers' section below I expand a little on the dangers of using the font-changing commands given in [[6, Section 3.1]].

Many of the more complicated Mathematical things in this tutorial are not documented in [[5]] or in [[6]]. The LaTeX team warns me that they feel no obligation to support commands that are not in [[6]], so there is a danger that some of these things may change. However, everything given here works, in both LaTeX 2.09 and in LaTeX 2ε, as at January 1995.

Some of the tricks described in this tutorial are at the edge of what you can conveniently do without using the `amstex` package. That package is undergoing change at the moment: I hope that by the time I reach the end of this sequence of tutorials the `amstex` package will have stabilized enough for someone to write an article explaining how to use it, including giving better methods than I can give here.

2 Answers

I promised to answer all questions arising from this series of articles (as far as I can).

2.1 Uneven subscripts

In *Baskerville* 4.5 Malcolm Clark asks about uneven baselines in subscripts. He gives a method of ensuring that all subscripts have the same baseline. I think that many Mathematical writers will not require that; nonetheless, some of us are uncomfortable with the unevenness in a single term such as

$$4z_1z_2^3$$

The easy way around this is to put a dummy superscript on the z_1, because it is the superscript on the z_2 that is pushing the 2 down: thus

`4 z_1^{} z_2^3` $4z_1^{}z_2^3$.

2.2 Roman text in notation

He also muses on whether to use `\textrm` or `\mathrm` or `\rm` in subscripts, if you are using LaTeX 2ε. My advice is never to use `\textrm` in Mathematical notation. In the first place, `\textrm` does *not* give you roman type, according to such expert references as [[1, 2, 3]], all of whom say that 'roman' type is upright, as opposed to italic. All that `\textrm` does is give you back serifs and proportional spacing, if you had turned them off. Perhaps he meant `\textup`. But, secondly, I don't think that you should use *any* of the commands `\text...` in Mathematical notation, because their effect depends on the surrounding text font but notation should be independent of the surrounding text. For example, try the following and compare the output:

```
{\rm $x_{\textup{big}} + \textup{size}_3$}
{\bf $x_{\textup{big}} + \textup{size}_3$}
{\bf $x_{\textrm{big}} + \textrm{size}_3$}
```

Malcolm was concerned because he wanted to obey the instruction in [[4]] to always use commands like `\textit{...}` rather than switches like `\it`. The trouble with that instruction is that the new commands `\text...` all work in a relative way. In my experience of writing (a lot of) Mathematics I have *never* needed such a relative change. I always need to specify my fonts absolutely, so that, say, the font chosen for long names of variables to be analysed does not change as the surrounding text font changes. Of course, it is sensible to do this with a macro such as `\variablename`; but that macro needs to call something with a syntax similar to `\textsl{...}` but which makes an absolute font change. I tried to persuade the LaTeX team to include commands like `\basesl{...}`, `\basett{...}` for such absolute changes, but I failed. Since the team wants to reserve the right to remove switches like `\tt` at some future time, this means that most of us will have to write our own macros, with our own idiosyncratic names, something like the following:

```
\DeclareTextFontCommand{\basett}%
    {\normalfont\ttfamily}
```

2.3 Spaces in subscripts

Malcolm also asked how to get spaces into subscripts. If I need to put a verbal phrase in a subscript then I use `\rm` and put in the interword spaces by hand.

```
\sum_{p{\rm\ is\ prime}} \frac{1}{p}
```

$$\sum_{p\rm\ is\ prime} \frac{1}{p}$$

2.4 Empty set

Kathleen Lyle has queried the symbol I gave last time for the empty set, with the command `\emptyset`. She points out that [[4]] shows a different symbol given by this command, a symbol which looks like a circle with a diagonal line through it and which is much closer to a Mathematician's idea of the empty set than is $\emptyset$. But [[4]] also gives the command `\varnothing`, available with the package `amssymb`, which produces the symbol $\emptyset$. It appears that Knuth made a mistake in using the name `\emptyset` for the glyph which most of us think of as a variant form of zero. To correct this mistake, the AMS has redefined the command `\emptyset` to produce the symbol more like the empty set and given us `\varnothing` for the sake of those authors who really do want a zero with a line through it. It is a pity that [[4]] does not say that its `\emptyset` is the AMS one rather than the Knuth one.

What to do when a software author makes a mistake like this is a controversial question. Personally, much as I prefer the AMS's empty set, I deplore such redefinition of a command, because it destroys portability of documents. Suppose that I write a document without the `amssymb` package and use `\emptyset`. I may send this document to someone (perhaps the AMS itself) who always uses the `amssymb` package when compiling documents. Even though I have made no explicit calls to commands defined by the package, my empty sets will come out looking different. A topologist may be content with the change; a computer scientist may not. In either case the document is printed with different symbols in the two cases, and this really should not happen. I think that it would have been better if the AMS had used a different name, such as `\trueemptyset`, for their empty set: then authors with access to the `amssymb` package could choose whether or not to include

```
\renewcommmand{\emptyset}{\trueemptyset}
```

at the start of their files.

5 A Spaced-out Interlude

5.1 Quads

Traditionally, there are certain lengths of space (depending on the type size) which are always used in certain places in Mathematical typesetting. The most useful are the *quad* space and the two-quad space. When I was a copy-editor I used to just put the marks for these two types of space in the appropriate places in the copy; I did not have to know how big they were. Neither do you. In displayed Maths, use `\qquad` to obtain a two-quad space between a formula and a short verbal condition or justification.

```
y \in Y \qquad\mbox{by defintion of~$Y$}
```

If there are two short formulas linked in a display by a short verbal phrase (perhaps only one word) use `\quad` to produce a quad space on either side of the phrase.

```
A \subseteq B \quad\mbox{and}\quad A \ne B
```

5.2 Other Spaces

A sequence of much smaller horizontal spaces that you can insert yourself is, in increasing order of magnitude,

```
\,  \:  \;  \␣
```

They are called *thin space*, *medium space*, *thick space* and *interword space* respectively; their size also depends on the current type size. The thin space is usually needed after the `!` in factorials and often needed after a square root.

`\sqrt{3} \, a`	$\sqrt{3}\,a$
`5!\,4!`	$5!\,4!$

It is also used before each *dx* term in an integral. On the other hand, in multiple integrals the integral signs may be too far apart, in which case the *negative* thin space `\!` should be inserted between them.

For consistency, these adjustments should all be made via macros. For example,

```
\newcommand{\sqrtsp}[1]{\sqrt{#1}\,}
```

will make `\sqrtsp` into the command for a square root with a little extra space, and a macro for factorials can be made similarly. For the integral signs you can use

```
\newcommand{\intt}{\int\!}
```

or the rather different solution provided in `amstex`. A suitable macro for the *dx* is

```
\newcommand{\diff}[1]{\, d #1}
```

which has the added advantage that if you believe that only variables should be in Maths italic then the `{\, d #1}` can be changed to `{\, {\rm d} #1}`.

5.3 Phantoms

The useful command `\phantom` allows you to leave a space whose horizontal and vertical dimensions are those of its argument. For example, if you want to define the notation $[]$ as the least-integer function without specifying a dummy variable, you can type `[\phantom{x}]`.

All digits are the same width, so `\phantom{0}` produces a phantom digit. It is very useful in tables of data when all other methods of alignment fail. Make yourself a macro for it.

There are also horizontal and vertical phantoms `\hphantom` and `\vphantom` respectively. Each of these measures only one dimension of its argument.

5.4 Horizontal Expanders

In the first tutorial we saw that `\widehat` and `\widetilde` expand as far as necessary (up to a given upper bound) to cover their arguments. The following commands also expand horizontally to match the arguments:

```
\overline         \underline
\overrightarrow   \overleftarrow
\overbrace        \underbrace
```

You can use a superscript to put a label on an overbrace, and a subscript with an underbrace.

```
n\bar{y}^2 +
\overbrace{(y_1-\bar{y})^2 + \cdots +
(y_n-\bar{y})^2}^{\rm sum\ of\ squares}
```

$$n\bar{y}^2 + \overbrace{(y_1-\bar{y})^2 + \cdots + (y_n-\bar{y})^2}^{\rm sum\ of\ squares}$$

6 Exercises

Exercise 23

$$(x_1+x_2)^3 = x_1^3+3x_1^2x_2+3x_1x_2^2+x_2^3$$

Exercise 24

$$\sum_{n \text{ divides } 10} n = 18$$

Exercise 25 In geometry, $\overrightarrow{AB}+\overrightarrow{BC}=\overrightarrow{AC}$.

Exercise 26 We define P_g by

$$t(vP_g) = (t^{g^{-1}})v \qquad \text{for } v \in \mathbf{R}^{\mathcal{T}}.$$

Exercise 27

$$2^a \times 2^b = \underbrace{2\times\cdots\times 2}_{a \text{ factors}} \times \underbrace{2\times\cdots\times 2}_{b \text{ factors}} = 2^{a+b}.$$

Exercise 28 An inner product $\langle\ ,\ \rangle$ is defined on G^* by

$$\langle\theta,\phi\rangle = \frac{1}{|G|}\sum_{g\in G}\theta(g)\overline{\phi}(g).$$

Exercise 29 If $\overline{}$ denotes complex conjugation, then

$$\overline{\xi+\zeta}=\overline{\xi}+\overline{\zeta} \quad \text{and} \quad \overline{\xi\zeta}=\overline{\xi}\,\overline{\zeta}.$$

Exercise 30

$$\int\!\!\int \phi(r,\theta)\,dr\,d\theta$$

Exercise 31

$$^6C_2 = \frac{6!}{4!\,2!}$$

7 Operators and relations

7.1 Limits

In the second tutorial I introduced various things that could have their limits, or ranges, typed in as sub- and superscripts: standard functions with English names, like `\log`; repeated binary operators, like `\sum`; and the integral sign `\int`. TEX thinks of all of these as `operator`s. Some operators have the limits set above and below in dispayed Maths, but to the right in text; others always have their limits set to the right. You can override these defaults by using one of the commands `\limits`, `\nolimits`, `\displaylimits` after the name of the operator. The integral sign normally has its limits set to the right: if you want them set above and below type `\int\limits`.

`\int_0^2 x^3 \, dx=4`	$\int_0^2 x^3\,dx=4$
`\int\limits_0^2 x^3 \, dx=4`	$\int\limits_0^2 x^3\,dx=4$

If you want the limits to be above and below if the operator happens to be in displayed Maths, but to the right otherwise, use `\displaylimits` instead of `\limits`. Finally, to ensure that the limits always come to the right, use `\nolimits`.

If you want to change the size of the operator as well as the position of its limits, you probably need to see the section on styles below.

7.2 Operators

The standard functions with English names already provided by TeX cannot be enough for the whole of Mathematics. You make new ones by using `\mathop`, usually inside a `\newcommand`. For example,

```
\newcommand{\var}{\mathop{\rm Var}\nolimits}
\var X \geq 0
```

$$\mathop{\rm Var}\nolimits X \geq 0$$

(If you have LaTeX 2_ε, you may feel safer using `{\mathrm{Var}}` in place of `{\rm Var}`.) You may put one of `\limits`, `\nolimits`, `\displaylimits` after the contents of the `mathop`, to specify how sub- and superscripts should behave. Putting nothing is equivalent to putting `\displaylimits`.

There is a school of thought that all operators should be in the same font, so that the `\rm` in the definition of `\var` should be replaced by a command like `\operatorfont`, which would, of course, be defined in the style file or in the preamble to the document. I do not agree with this. It is not at all unusal to use bold for the expectation operator while retaining roman for the variance.

If you make a single letter into an `operator`, it will be vertically centred, which may not be what you intend:

```
\newcommand{\ee}{\mathop{\rm E}\nolimits}
\[\ee X + \ee Y = \ee(X+Y)\]
```

$$\mathop{\rm E}\nolimits X + \mathop{\rm E}\nolimits Y = \mathop{\rm E}\nolimits(X+Y)$$

To override this, put the single letter in a box:

```
...\mathop{\mbox{\rm E}}...
```

7.3 Novel uses of operators

In the first tutorial I said that you did not need to think of the symbol `'` as a superscript. Usually you do not, but TeX always does, so you occasionally get unexpected results. You might want to write $\sum'$ for a variant of the usual summation, perhaps to indicate omission of all i for which $\lambda_i = 0$, as in

$$\mathop{\sum\nolimits'}_{i=1}^{n} \frac{1}{\lambda_i} P_i.$$

If you use `\sum'` it will come out as

$$\sum'$$

in display, and even worse things happen when you try to put limits on. Writing `\sum\nolimits'` cures the problem about placing the dash, but then you no longer have an `operator` to put limits on. So you need to make the whole of $\sum'$ into an `operator`:

```
\newcommand{\summ}{\mathop{\sum\nolimits'}}
\[\summ_{i=3}^{7}\]
```

$$\mathop{\sum\nolimits'}_{i=3}^{7}$$

If you look closely you will now see that the limits are centred on the whole of $\sum'$. This is logical, but may not be exactly what you intended. I do not know how to do the illogical but more aesthetically pleasing version, but a method is provided in `amstex`.

Sometimes you want to put a range of summation under (or over) the middle of a pair of summation signs. Do this by turning the pair of summation signs into an `operator`:

```
\newcommand{\twosum}{\mathop{\sum\sum}}
\[\twosum_{1<i<j<n} x_i x_j\]
```

$$\mathop{\sum\sum}_{1<i<j<n} x_i x_j$$

To get two ranges of summation under a summation sign, make an `operator` containing the summation sign and the interior range(s), and then put a subscript on that:

```
\[\mathop{\sum_{j=1}^{n}}_{j\ne i} Y_j\]
```

$$\mathop{\sum_{j=1}^{n}}_{j\ne i} Y_j$$

You would normally do this only in displayed Maths.

7.4 Binary operators

TeX does not class ordinary binary operators as `operator`s. Use `\mathbin` to make something into an infix binary operator. For example, `$n\mathbin{**}r$` gives $n \mathbin{**} r$. (What does `$n**r$` produce?) Usually this is done with a `\newcommand`. Even a single symbol may need to be explicitly turned into a `mathbin`, if it is not one already, so that the spacing and linebreaks around it are correct: this is as true for single symbols that already exist as for those that you build up laboriously out of pieces. If the new binary operator is (part of) an English word, you will need to specify the font, just as for `\mathop`.

7.5 Binary relations

In the same way, `\mathrel` is used to make new binary relations. The considerations are similar to those for `mathbin`s. Note that `mathbin`s and `mathrel`s are different in several subtle ways, such as the spacing around them, the linebreaks near them, and their behaviour when they do not find themselves between two ordinary symbols (compare `$n**r$` with `$n==r$`). If you are not a Mathematician you will probably have to ask the author of the document whether a particular squiggle is an operator or a relation.

If the new relation consists of two parts, one on top of the other, you can make the new relation in one step with `\stackrel`.

`\Phi\stackrel{\rm rev}{\sim}\Psi` $\Phi\stackrel{\rm rev}{\sim}\Psi$

7.6 Styles

When `\sum` appears in displayed Maths it is larger than in text Maths, and has its limits in a different place. However, once it is inside a fraction or an array, even in displayed Maths, it reverts to its appearance in text Maths. To force one style or the other, precede `\sum` with either `\displaystyle` or `\textstyle`.

`\[\frac{\sum_i x_i}{n}\]` $$\frac{\sum_i x_i}{n}$$

```
\[\frac{\displaystyle\sum_i x_i}{n}\]
```

$$\frac{\displaystyle\sum_i x_i}{n}$$

These two commands can affect the appearance of many items in Maths mode, including `\frac`.

There are analogous commands `\scriptstyle`, which sets the following items as if they were in a subscript, and `\scriptscriptstyle`, which sets them as if they were in a second-level subscript.

None of these four commands takes an argument. They are all switches, like `\rm` and `\large`, and apply until the end of the current subformula (such as the numerator of a `\frac`).

8 Exercises

Exercise 32 If f is a probability density function then

$$\int\limits_{-\infty}^{\infty} f = 1.$$

Exercise 33 We assume that Y is a random vector with

$$\operatorname{Cov} Y = \sum_{\alpha\in A} \xi_\alpha S_\alpha,$$

where the S_α are known symmetric matrices satisfying $S_\alpha S_\beta = \delta_{\alpha\beta} S_\alpha$ and $\sum_{\alpha\in A} S_\alpha = I$.

Exercise 34 The definition of variance is: $\operatorname{Var} X = \mathbf{E}\,(X - \mathbf{E}\,X)^2$.

Exercise 35 The optimize function opt is defined so that $\operatorname*{opt}_{i=1}^{n} M_i$ is equal to $\max\{\max_i M_i, 0\}$.

Exercise 36

$$\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} y_i y_j = \left(\sum_{i=1}^{n} y_i\right)^2 - \sum_{i=1}^{n} y_i^2.$$

Exercise 37

$$\sum_{\substack{0<i<m\\ 0<j<n}} P(i,j).$$

Exercise 38 Define the operator $\boxed{+}$ on finite sets of integers as follows. If A and B are two finite sets of integers, then $A \boxed{+} B$ is the multiset of integers in which the number of times that n occurs is equal to

$$|\{(a,b) : a \in A,\ b \in B,\ a+b=n\}|.$$

Exercise 39 We want to write our wreath products in reverse order, so we put $G \operatorname{rw} H = H \wr G$.

Exercise 40 The relation ρ is said to be symmetric if

$$x \mathbin{\rho} y \quad \text{implies} \quad y \mathbin{\rho} x.$$

Exercise 41 The strong law of large numbers states that if $X_1, X_2, \ldots$ are independent and identically distributed with finite fourth moment then

$$\frac{X_1 + \cdots + X_n}{n} \xrightarrow{\text{a.s.}} Y,$$

where $\Pr[Y = E(X_1)] = 1$.

Exercise 42 Define the relation $<_2$ on the natural numbers by $n <_2 m$ if $n \mid m$ and m/n is odd. This is a partial order.

Exercise 43 *Create a binary operator for the colon in $G\!:\!H$ and compare it with ':' and* `\colon`.

Exercise 44 (*Redo Exercise 16 with a built-up fraction instead of the solidus, with the large operators remaining the same size.*)

$$\prod_{k\geq 0}\frac{1}{(1-q^k z)} = \frac{\sum_{n\geq 0} z^n}{\prod_{1\leq k\leq n}(1-q^k)}$$

Exercise 45 *Redo Question 41 with the 'a.s.' in normal-sized type.*

Exercise 46 More on binomial coefficients:

$$\sum_{\substack{1\leq n\leq m\\ 1\leq r\leq n}} \frac{n!}{r!\,(n-r)!} \quad = \quad \sum_{n=1}^{m} 2^n \quad = \quad 2^{m+1} - 2.$$

Exercise 47

$$\prod_{i=0}^{m}{}^{*} f(\lambda_i)$$

References

[1] *Hart's Rules for Compositors and Readers*, Oxford University Press, Oxford, (1967).

[2] *The Chambers Dictionary*, Chambers Harrap, Edinburgh, (1993).

[3] *The Chicago Manual of Style*, The University of Chicago Press, Chicago, (1982).

[4] Goossens, M., Mittelbach, F. & Samarin, A.: *The LaTeX Companion*, Addison-Wesley, Reading, Mass., (1994).

[5] Lamport, L.: *LaTeX: A Document Preparation System*, first edition, Addison-Wesley, Reading, Mass., (1986).

[6] Lamport, L.: *LaTeX: A Document Preparation System*, second edition, Addison-Wesley, Reading, Mass., (1994).

XII Maths in LaTeX: Part 4, Numbered and Unnumbered Things

R. A. Bailey
Queen Mary and Westfield College,
University of London

Recall

This is the fourth in a sequence of tutorials on typesetting Mathematics in LaTeX. The first three appeared in issues 4.4, 4.5 and 5.1 of *Baskerville*. The series includes some things which can be found in [4], but I am working in more things which, while straightforward and necessary for Mathematical work, are not in [4] or [5]. In this tutorial I concentrate not on Mathematical formulae but on things like equations and theorems which Mathematicians like to display in special ways and to number (or not).

In case you missed the first three tutorials, I remind you that I expect you, the reader, to do some work. Every so often comes a group of exercises, which you are supposed to do. Use LaTeX to typeset everything in the exercise except sentences in italics, which are instructions. If you are not satisfied that you can do the exercise, then tell me. Either write to me at

School of Mathematical Sciences
Queen Mary and Westfield College
Mile End Road
London E1 4NS

with hard copy of your input and output, or email me at `r.a.bailey@qmw.ac.uk` with a copy of the smallest possible piece of LaTeX input file that contains your attempt at the answer. In either case I will include a solution in the following issue of *Baskerville*: you will remain anonymous if you wish.

Answers

I promised to answer all questions arising from this series of articles (as far as I can).

Size of parentheses

Charles M. Goldie asks why I put $(t^{g^{-1}})v$ in Exercise 26 instead of using `\bigl(` and `\bigr)` to make the parentheses larger than what they enclose. The answer is that you would need `\Bigl(` and `\Bigr)` to make them large enough in this case: I should have used `\left` and `\right` (see Part 2 of this series) but I was lazy.

Both he and Charles Leedham-Green have asked why I permit, or even encourage, deeply nested parentheses, as in

$$u(P((1-\varepsilon)z)), \tag{1}$$

without using commands like `\bigl(` to make some of the outer parentheses somewhat larger. In Chapter 17 of [3], Knuth advises that authors should use `\bigl` and its relatives to specify the size of parentheses and other expandable fences, to improve the readability of their formulae. However, I have deliberately avoided telling people about these commands.

I have two reasons for ignoring Knuth's advice. The first is that I regard LaTeX primarily as a system for *authors*, not for typesetters. Authors should not be stopping to worry about the size of parentheses, particularly if the level of nesting may change in a later version of the document. Conventions on size of fence should be a matter for the style designer, not the author. If someone can write a style file that automatically detects the level of nesting and adjusts the size accordingly, well and good. I have no objection to using such a style file; I do object to interrupting my Mathematical thoughts to fret over niceties of sizing.

My second reason is related to the first. Journal editors and executive editors tend to have policies about the size of fences, and they will impose these policies whatever we as authors do. So it is generally a *complete waste of time* for authors to use `\Bigr` and the like, or for referees to insist on them.

This is not to say that I disapprove of `\left` and `\right`. These commands automatically adjust the size of the fence to fit what is inside it. Adding, say, an extra item inside a `\left( ... \right)` does not cause the author to rethink the size of the parentheses. In fact, in my basic style file I have macros such as `\probab`, `\setof` and `\card` which use `\left` and `\right` precisely so that I can type as I think, *the set of* ... rather than *squiggly brackets, now what size and how much space?* (Oh, all right:

```
\newcommand{\card}[1]{\left|#1\right|}
```

—you can guess the others.) In fact, `\left` and `\right` make no difference to the formula in (1), so neither of my questioners will be satisfied by my answer.

Interchangeability of parentheses, brackets and braces

Charles M. Goldie also asks if I have an opinion about whether nested parentheses should be routinely replaced so that one uses the sequence $\{[(\ldots)]\}$, which is demanded by some journals. I do have an opinion, quite a strong one, probably because one of the journals in which I publish most frequently insists on the sequence $[\{(\ldots)\}]$ and shows surprise (or the executive editor does) each time that I explain that I am using $\{\ldots\}$ to denote a *set*. My opinion has been admirably expressed by Ellen Swanson in her bible of Mathematical typesetting [6]:

> Often, however, the author of research mathematics attaches a special meaning to different types of enclosures, and this author believes that they *should be left in whatever order and variety the author has indicated in the manuscript.*

(her italics).

5 Numbered and Unnumbered Displays

5.1 Unnumbered Maths displays

Use `\[` and `\]` for an unnumbered single line of displayed Maths: see Part 1. If you have two or more lines of displayed Maths that must be vertically aligned then you need one of the `array` environments. I shall deal with them in the final tutorial in this series.

5.2 Unnumbered word displays

Sometimes what you want to display is not simply a formula but a verbal condition that may or may not involve short pieces of notation. For example:

> each basis vector f in $V_{T,B}$ is orthogonal to every basis vector in $V_{B,T}$ except $f\psi$.

If this will fit on a single line then you can use `\[\mbox{...}\]`, but this is not very satisfactory because you have to stop and think how long it is and it is subject to changes in the line width. I find that the `quote` environment works well for such displays.

5.3 Numbered Maths displays

Use the `equation` environment for a numbered single line of displayed Maths such as (1) in the 'Answers' section above. If you have two or more consecutive equations or formulae that do not need to be vertically aligned, simply use one `equation` environment per line. For vertical alignment, wait until the tutorial on arrays.

If you put a `\label` within an `equation` you can painlessly refer back (or forward) to that equation.

```
For contrasts, we put
\begin{equation}
W_T = V_T \cap V_0^\perp
\label{contrasts}
\end{equation}
The space~$W_T$ was defined
in Equation~(\ref{contrasts}).
```

> For contrasts, we put
> $$W_T = V_T \cap V_0^\perp \tag{2}$$
> The space W_T was defined in Equation (2).

5.4 Numbered word displays

Sometimes word displays also need to be numbered for reference. LaTeX does not directly provide an environment for this, but I find that the following works quite well.

```
There is a bijection ...
\begin{equation}
\begin{minipage}[t]{0.8\linewidth}
```

```
each basis vector~$f$ ...
\end{minipage}
\end{equation}
Using ...
```

There is a bijection ψ between the bases which satisfies:

each basis vector f in $V_{T,B}$ is orthogonal to every basis vector in $V_{B,T}$ except $f\psi$. (3)

Using ψ, we can show that ...

These displays are numbered in the same sequence as the `equation`s, and can be labelled and referred to in just the same way. Note that I have made no attempt to make the indentation the same as that in `quote`.

You can suppress the `[t]` if you want the number to be vertically centred on the display. You can replace 0.8 by any reasonable fraction. There is a catch, however: if you have numbered word displays labelled (9) and (10) you may find that the second one comes out shifted to the left, to allow space for the wider label. Whether or not this happens depends on the settings of other parameters, such as `\linewidth`. With the default width for A4 paper in 10pt in LaTeX 2ε, I found that I had to decrease 0.8 to 0.75 in order to have satisfactory word displays numbered (9) and (10).

Of course, if you have two or more such displays you should make an environment for them. I do it as follows.

```
\newenvironment{condition}%
 {\equation%
  \begin{minipage}[t]{0.8\linewidth}}%
 {\end{minipage}\endequation}
```

You may wonder why I have used `\equation` and `\endequation` in the definition instead of `\begin{equation}` and `\end{equation}`. This is because of the clever things that LaTeX does with spaces before and after displayed material. When you type the line

```
\end{equation}
```

LaTeX ignores the spaces on the rest of the line; if you type the line

```
\end{condition}
```

and the final part of the `condition` environment is `\end{equation}` then this forgetfulness about spaces is not passed through to `\end{condition}`. Use of the more primitive `\equation` and `\endequation` does pass on the forgetfulness.

5.5 Numbering equations within sections

By default, `equation`s are numbered 1, 2, ... right through the document in the `article` class. To make them numbered within sections you need

```
\renewcommand{\theequation}%
  {\thesection.\arabic{equation}}
```

Then the first equation in Section 1 will be numbered 1.1, the next 1.2, and so on. However, if there are four equations in Section 1, then the first equation in Section 2 will be numbered 2.5 because the `equation` counter has not been reset at the start of the new section. To correct this, you also need

```
\@addtoreset{equation}{section}
```

Because of the `@` sign in this command, you must either place it in a style file or make sure that it comes between the commands `\makeatletter` and `\makeatother` in the preamble to the document.

5.6 One-off numbering of equations

Occasionally you want to number an equation not in the main sequence but by a particular symbol, such as $(*)$ or $(1.1')$. Use the following `oneoff` environment in place of `equation`, putting the desired symbol as the single parameter.

```
\newenvironment{oneoff}[1]{\equation%
  \addtocounter{equation}{-1}%
  \renewcommand{\theequation}{\mbox{#1}}}%
  {\endequation}
```

For example,

```
\begin{oneoff}{$*$}
a(p_i,q) - a(p_j,q) = 0 \bmod s
\end{oneoff}
```

$$a(p_i,q) - a(p_j,q) = 0 \bmod s \tag{$*$}$$

(See [4, page 92] or [5, pages 98–99] for how these counter commands work.) Note that automatic cross-referencing does not work for such equations.

If you want a one-off equation numbered 1.1′ related to Equation (1.1) then give the latter a label (say, `rowsum`) and then do

```
\begin{oneoff}{\ref{rowsum}$'$} ...
```

5.7 Subsequences of equations

Suppose that between Equations (5) and (7) you want a sequence of equations numbered (6a), (6b) etc. Put the following in the preamble to the document (or in the style file).

```
\newsavebox{\saveeqn}
\newcounter{subeqnno}
\renewcommand{\thesubeqnno}{\alph{subeqnno}}
\newenvironment{subequations}%
    {\refstepcounter{equation}%
     \savebox{\saveeqn}{\theequation}%
     \setcounter{subeqnno}{0}}%
    {}
\newenvironment{subeqn}%
    {\refstepcounter{subeqnno}%
     \oneoff{\usebox{\saveeqn}\thesubeqnno}}%
    {\endoneoff}
```

(See [4, page 101] or [5, pages107–108] for details of `\savebox`.) Then use `subeqn` in place of `equation` for each of the equations (6a), (6b) etc., and place the whole subsequence in the `subequations` environment.

Automatic cross-referencing doesn't work for these either. The reason is that in constructing `oneoff` and `subeqn` I have *used* the `equation` environment rather than *mimicking* it, with the result that any `\label` picks up the `equation` counter. To do this properly you would have to copy out the `equation` part of `latex.tex` (which is well documented) and hack it (for 2.09ers; of course, LaTeX 2_ε persons would have to hack part of `ltmath.dtx` and `classes.dtx`, which some would argue are even better documented). I have never needed this construct often enough to bother to do it properly, but I am sure that it could be done.

Exercises

Exercise 48 *Make a numbered displayed equation saying*

$$t\,(vP_g) = \left(t^{g^{-1}}\right)v$$

and a sentence which refers to it.

Exercise 49 *Make an unnumbered word display saying*

There is a natural surjective homomorphism $\phi\colon G \to G/N$ with $\ker(\phi) \simeq \mathrm{Im}(\phi)$.

Exercise 50 *Make a displayed numbered verbal condition saying*

for all A, B, C in $\mathcal{P}$: if $A \prec B$ and $B \prec C$ then $A \prec C$; and if $A \preceq B$ and $B \preceq A$ then $A = B$.

Then add a sentence which refers to it.

6 Theorems and their friends

6.1 Basics

To make a new environment called `thm` for theorems, do

```
\newtheorem{thm}{Theorem}
```

This sets up the environment, which is then used as follows.

```
\begin{thm}
The kernel of a homomorphism
is a congruence.
```

```
\label{basic}
\end{thm}
In Theorem~\ref{basic} we ...
```

Theorem 1 *The kernel of a homomorphism is a congruence.*
In Theorem 1 we ...

The theorems are all given the heading 'Theorem'. They are numbered automatically, and may be cross-referred to in the usual way.

For clarity in the rest of this section, I shall call the item like `thm` the *theorem environment*, the item like `Theorem` the *theorem name*, the text like 'Theorem 1' the *theorem head*, and the text like 'The kernel of ... ' the *theorem body*.

Note that there is nothing to prevent two different theorem environments having the same theorem name. Indeed, the theorem name can be empty.

By default, the theorem head is in bold and the theorem body is in italics. The theorems are numbered in arabic numbers, in a single sequence throughout the document (in the `article` class). All of these defaults can be changed, as I show below.

It may not be obvious to the novice user, but there is more to an environment created with `\newtheorem` than special layout and automatic numbering. The spacing before and after each theorem environment is controlled, and penalties are set so that no page break will come after the first line of a theorem environment unless there is a natural break-point in the text.

Unfortunately, there is a bug (oops, feature) in LaTeX which means that if you put `\label{...}` immediately after `\begin{thm}` you spoil this page-breaking penalty. But the `\label` should be somewhere easy to find, so I always play safe and put it immediately before `\end{thm}`.

6.2 Named theorems

If you have a theorem environment `thm` then you can use an optional argument to `thm` to obtain a named theorem. For example,

```
\begin{thm}[The Central Limit Theorem]
If $X_1$, ...
```

Theorem 2 (The Central Limit Theorem) *If $X_1, X_2, \ldots, X_n$ are independent ...*

or

```
\begin{thm}[Galois, 1832]
```

Theorem 3 (Galois, 1832) *If $L : K$ is a finite normal ...*

6.3 Sequences of numbering

Two optional arguments to `\newtheorem` give you control of which theorem environments are numbered in which sequences. Although it is logical to number theorems, lemmas, corollaries etc. all in their own sequences, it is much easier to find your way around a long document if they are all in a single sequence. To get a theorem environment `lem` numbered in the same sequence as `thm`, do

```
\newtheorem{lem}[thm]{Lemma}
```

After the theorems we have had so far, if we now do

```
\begin{lem}
With the above notation ...
\end{lem}
```

we get

Lemma 4 *With the above notation ...*

The other optional argument numbers the theorem environment inside something else. If you want the second example in Section 3 to be numbered 3.2 irrespective of how many examples there were in previous sections, then do

```
\newtheorem{eg}{Example}[section]
```

You can use at most one optional argument with each `\newtheorem` command.

You can even number one theorem environment inside another: for example

```
\newtheorem{cor}{Corollary}[thm]
```

if you want the corollaries after Theorem 10 to be Corollary 10.1, Corollary 10.2, etc. Be careful not to create a circle of environments numbered within each other.

6.4 Unnumbered environments

There are several items, such as definitions, remarks and notation, that clearly should be theorem-like environments except that they should not be numbered. It would be possible to set them all up and then separately adjust the counter on each one so that it is not numbered. However, it is easier to take advantage of the number-in-the-same-sequence option. Set up a single unnumbered counter with

```
\newcounter{unnumber}
\renewcommand{\theunnumber}{}
```

and then put the other unnumbered theorem environments in the same sequence with

```
\newtheorem{rem}[unnumber]{Remark}
\newtheorem{def}[unnumber]{Definition}
```

6.5 Other systems of numbering

Many Mathematicians want the possibility of having Theorem A, Theorem B etc. as well as Theorem 1 etc. This is no problem. Use the commands given in [4, page 92] or [5, page 98] to alter the way a theorem environment is numbered. Thus

```
\newtheorem{thma}{Theorem}
\renewcommand{\thethma}{\Alph{thma}}
\begin{thma} The subgroups ...
...
\begin{thma} The irreducible  ...
\label{char}
...
The result of Theorem~\ref{char} \ldots
```

Theorem A *The subgroups* . . .

Theorem B *The irreducible* . . .

The result of Theorem B . . .

Other possibilities for numbering are

`\alph{thma}`	Theorem b
`\roman{thma}`	Theorem ii
`\Roman{thma}`	Theorem II

6.6 Changing the fonts

Fonts are handled differently in LaTeX 2.09, in NFSS, and in the new standard LaTeX, LaTeX 2ε. All the suggestions that I give in this section work in both LaTeX 2.09 and LaTeX 2ε. They do not work at all if you run NFSS without LaTeX 2ε. If you are using LaTeX 2ε, you *must* use the forms like `\sc` given here: the commands like `\textsc` will not do the right thing, because they *add* small capitals (say) to the default fonts instead of *replacing* the default fonts.

The `\newtheorem` command in LaTeX is the most wonderful thing to happen to Mathematical writers in a long time, because so many of our constructs fit it. However, one of the worst things to happen to Mathematical writers is the hard-wiring of the fonts for the theorem heads and the theorem bodies. Ordinary LaTeX simply does not give you the flexibility to change these easily. Yet the defaults are not always appropriate, and different journals demand different fonts for these purposes. I suspect that this hard-wiring is one reason that some Mathematicians have been reluctant to use LaTeX. What can the ordinary user do about this problem?

I shall give four answers, because different solutions are appropriate in different circumstances.

(i) *Bare hands* Sometimes (for example, when sending an article to *Baskerville*) you cannot submit your favourite style files along with your main file. So you need a ‘bare hands’ way of changing the fonts, without losing too much genericity. Here’s how.

To make a theorem environment `prop` whose theorem head is set in small capitals, do

```
\newtheorem{prop}{\sc Proposition}
```

For a small document, this will do. To be more generic, you could do

```
\newcommand{\headfont}{\sc}
\newtheorem{prop}{\headfont Proposition}
```

so that only one line has to be changed if you decide to change the font of all the theorem heads.

To make a theorem environment `qn` whose theorem body font is set in roman, use a two-stage process. The exercises in this sequence of tutorials are defined by

```
\newtheorem{preqn}{Exercise}
\newenvironment{qn}{\preqn\rm}%
                  {\endpreqn}
```

(The exercises in this particular tutorial are bad examples, because they all have italic instructions.)

(ii) *Mittelbach's style file* Frank Mittelbach wrote the style file `theorem.sty`, which should be available from all good CTAN hosts. It is described in [2, pages 251–255]. It enables you to change the fonts and the layout of theorem environments. However, it does not do exactly what I need.

(iii) *My style file* When I first started to use LaTeX it was obvious to me that a flexible means of changing the fonts for theorem heads and theorem bodies had to be provided. I hacked `@begintheorem` from `latex.tex` to produce a style file which lets me give a single command to set the font for theorem heads, another to set the font for (most) theorem bodies, and another to say that all subsequently declared theorem environments will have their bodies set in 'ordinary' type: not necessarily in roman, but in *whatever font the surrounding text is in*. However, like many others, I deplore the proliferation of personal style files because they inhibit portability of documents, so I haven't made this style file generally available.

(iv) *American Mathematical Society* The old version of `amstex.sty` (see [1]) gives the user the possibility of declaring theorem environments in three classes—plain theorems, definitions and remarks. However, it does not give the user any control over the fonts used in those environments. I do not know if the forthcoming package `amsthm.sty` will give any more freedom.

6.7 Proofs

We all need a proof environment, so everyone invents her own, some more successfully than others. I think that the proof environment should be made with `\newtheorem`, so that all the benefits of spacing, of page-break penalties and of consistent head fonts can be retained. Of course, proofs should be unnumbered and (usually) set in the same font as the surrounding text. So I simply use the foregoing methods to create an environment `pf` with name `Proof` which is unnumbered and has its body set either in the surrounding text font or in roman.

What should you do about the end-of-proof symbol? Some people want it put in automatically. In principle this could be done with something like

```
\newenvironment{proof}{\pf}%
                      {\eop\endpf}
```

where `\eop` is your favourite end-of-proof symbol, for example

```
\unskip\protect\nolinebreak\mbox{\quad$\Box$}
```

This is not really satisfactory if you have any proofs that end in displayed Maths (or any other sort of display). Traditionally the end-of-proof sign goes in the display, not on a new line; but if you have a display inside an environment then LaTeX finishes off the display and gets ready for a new line before it reads the instructions for the end of the environment. So my advice is to have an `\eop` macro and put it in by hand at the end of every proof, either just inside the final display (if this is the last thing in the proof) or just before the `\end{pf}`.

6.8 Questions and Exercises

A theorem environment is ideal for questions on exam papers and coursework sheets, and exercises in text books. It is usually better than `enumerate` because it retains the normal textwidth, paragraph indentation and paragraph separation. If you want the questions to be headed simply '1', '2', etc. then do

```
\newtheorem{question}{}
```

If the questions have parts and subparts, it is sensible to use `enumerate` for them. In that case you probably need to change the default numbering of the `enumerate` environments so that, say, parts are labelled '(a)' etc. and subparts '(ii)' etc. The next section shows how to do this.

Exercises

Exercise 51 *Create a short document with two sections. In the first section put one theorem, a lemma subtitled 'Burnside', another theorem, and a remark. The remark should be neither numbered nor in italic. In the second section put another lemma, another theorem, a corollary numbered in the same sequence as the theorems, and finally a theorem in a roman-numbered sequence. Include cross-references to all the numbered items.*

Exercise 52 *Redo the previous question, in such a way that lemmas and equations are numbered within sections.*

7 Other numbered things

7.1 Numbered lists

If you use `enumerate` within a theorem environment then you will probably have to change the way that the different levels of enumerated list are numbered. This is controlled by commands containing the strings `enumi`, `enumii`, `enumiii` and `enumiv`. Thus the Nth level of nesting is controlled by `enumN`.

The counter for `enumN` is called simply `enumN`. To alter whether the counter is displayed as an arabic numeral, a letter etc., you change `\theenumN` (see [4, pages 91–92] or [5, pages 97–99].) To alter the printed labels which are put on the items in the list, change `\labelenumN` to be a suitable text containing `\theenumN`.

I find that two levels of nesting are quite sufficient within exam questions and homework problems. My style files for exams and homeworks contain the lines

```
\renewcommand{\theenumi}{\alph{enumi}}
\renewcommand{\labelenumi}{(\theenumi)}
\renewcommand{\theenumii}{\roman{enumii}}
\renewcommand{\labelenumii}{(\theenumii)}
```

In a book, you might need to put something similar in the start of an `exercises` environment.

If you are lazy then you might try to alter just `\labelenumN`. The list items will have the correct printed labels but your printed cross-references will not match. The cross-reference generated by a `\ref` call to a `\label` in the Nth level of nested `enumerate`s has the form

```
\p@enumN\theenumN
```

where `\p@enumN` usually picks up the `\theenumM` from higher levels ($M < N$), and possibly some punctuation. If you don't like the settings of `\p@enumN` that LaTeX gives you by default, you will have to change them in a style file.

7.2 Footnotes

Mathematicians usually don't use footnotes, because the footnote marks would be interpreted as superscripts or operators. However, we do sometimes like to put information at the bottom of the first page of a document, under a horizontal line: perhaps an address for correspondence, or a list of AMS subject categories. You can do this with a `\footnote` early in the document, so long as you have first done

```
\renewcommand{\thefootnote}{}
```

It is best to put this command in a small group around the use of `\footnote`.

Exercises

Exercise 53 *Modify the document in Exercise 51 so that one of the theorems has parts and subparts. The parts should be labelled*

[A], [B], ...

and the subparts

1/, 2/, ...

Exercise 54 *Modify the document in Exercise 53 so that the foot of the first page carries the text*

Key words: construction of designs; neighbour balance; optimality; randomization; software.

References

[1] AMERICAN MATHEMATICAL SOCIETY: *AMS-LaTeX Version 1.0 User's Guide*, American Mathematical Society, Providence, Rhode Island, (1990).

[2] GOOSSENS, M., MITTELBACH, F. & SAMARIN, A.: *The LaTeX Companion*, Addison-Wesley, Reading, Mass., (1994).

[3] KNUTH, D. E.: *The TeXbook*, Addison-Wesley, Reading, Mass., (1984).

[4] LAMPORT, L.: *LaTeX: A Document Preparation System*, first edition, Addison-Wesley, Reading, Mass., (1986).

[5] LAMPORT, L.: *LaTeX: A Document Preparation System*, second edition, Addison-Wesley, Reading, Mass., (1994).

[6] SWANSON, E.: *Mathematics into Type*, revised edition, American Mathematical Society, Providence, Rhode Island, (1979).

VII Maths in LaTeX: Part 5, Getting started on arrays

R. .A. Bailey
Queen Mary and Westfield College
University of London

Recall

This is the fifth part of a sequence of tutorials on typesetting Mathematics in LaTeX. The first four appeared in issues 4.4, 4.5, 5.1 and 5.2 of *Baskerville*. The series includes some things which can be found in [3], but I am working in more things which, while straightforward and necessary for Mathematical work, are not in [3] or [4]. In this tutorial and the next I cover arrays: matrices, tables of data, aligned equations, and other items with a two-dimensional layout.

I do not cover the extra array goodies provided by packages such as Frank Mittelbach's `array.sty`, David Carlisle's plethora of array add-ons or the new packages from the American Mathematical Society. I hope that someone else will write a follow-on article describing some or all of those, particularly if they can give easier methods of doing some of the more cumbersome items in this tutorial.

In case you missed the previous tutorials, I remind you that I expect you, the reader, to do some work. Every so often comes a group of exercises, which you are supposed to do. Use LaTeX to typeset everything in the exercise except sentences in italics, which are instructions. If you are not satisfied that you can do the exercise, then tell me. Either write to me at

School of Mathematical Sciences
Queen Mary and Westfield College
Mile End Road
London E1 4NS

with hard copy of your input and output, or email me at `r.a.bailey@qmw.ac.uk` with a copy of the smallest possible piece of LaTeX input file that contains your attempt at the answer. In either case I will include a solution in the following issue of *Baskerville*: you will remain anonymous if you wish.

8 Mathematical arrays

8.1 Basics

The `array` environment is used to set all manner of aligned Mathematics. A simple example is:

```
\begin{array}{lrrr}
x & 1 & 2 & 5\\
x^2 & 1 & 4 & 25 \\
\end{array}
```

$$\begin{array}{lrrr}
x & 1 & 2 & 5\\
x^2 & 1 & 4 & 25
\end{array}$$

This array has four columns. The alignment of these columns is shown in the argument `lrrr` of the environment. The first column is left-aligned; the other three are right-aligned. It is also possible to have a centred column, shown by a `c`. Typically columns of Mathematical symbols are centred, integers are right-aligned and powers of a single symbol are left-aligned. >From now on I shall refer to the argument such as `lrrr` as the *columns specifier*.

Entries in each row of the `array` are separated by ampersands. The end of a row (except the last one) is shown by `\\`. If the last few entries in a row are blank there is (usually) no need to put in all the ampersands.

The `array` environment can be used only in Maths mode. All its entries are automatically in Maths mode, in `\textstyle`. If you have fractions or summations in an array you may want to precede them by `\displaystyle`.

Each entry is in a separate 'box', which is typeset before the array is built up. Thus switches such as `\bf`, `\displaystyle` and `\raggedright` extend no further than the end of the current entry. If a binary relation or operator, such as $=$ or $+$, occurs at the beginning or end of an entry (LA)TeX cannot give it the proper spacing relative to the previous or succeeding entry.

$$\left(\begin{array}{rrr} 2 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 4 \end{array}\right)\left(\begin{array}{r} 5 \\ -3 \\ 8 \end{array}\right)=\left(\begin{array}{r} 5 \\ -22 \\ 30 \end{array}\right)$$

Figure 1. A matrix equation

$$\det\left[\begin{array}{cccc} \dfrac{\partial h_1}{\partial y_1} & \dfrac{\partial h_1}{\partial y_2} & \cdots & \dfrac{\partial h_1}{\partial y_n} \\ \dfrac{\partial h_2}{\partial y_1} & \dfrac{\partial h_2}{\partial y_2} & \cdots & \dfrac{\partial h_2}{\partial y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial h_n}{\partial y_1} & \cdots & \cdots & \dfrac{\partial h_n}{\partial y_n} \end{array}\right]$$

Figure 2. The Jacobian of *h*

8.2 Matrices and vectors

The most obvious use for `array` is for a matrix. Use `\left` and `\right` to surround the matrix with fences of the right size. For example, the matrix in Figure 1 begins

```
\left(
\begin{array}{rrr}
```

The commands `\vdots` and `\ddots` are useful in matrices. They are both used in Figure 2.

An array can have a single column, as in the column vectors in Figure 1, or a single row, which is useful for a permutation or a row vector.

8.3 Arrays of arrays

Try to think logically, rather than visually, about the contents of an array. For example, if you have six 5×5 Latin squares set out in a 2×3 rectangle then you might think that you have 10 rows and 15 columns. However, it is more logical to do `\begin{array}{ccc}` and then make each entry in this array a new array with five centred columns. One advantage of this approach is that it makes editing the file much easier when you decide to change the order of the Latin squares, or to lay them out in a 3×2 rectangle.

8.4 Changing the row spacing

The default spacing between the rows of an `array` is fine for simple matrices with numbers in, but is often insufficient for large entries, particularly if you use `\displaystyle`. To adjust the amount of space after the current row, put an explicit length in square brackets after the `\\`: for example, `\\[20pt]`.

I find it useful to work in multiples of `\jot`: this length is the usual extra space between lines of displayed Mathematics. Curiously, `\jot` is equal to 3 points in all of LaTeX's three point sizes, so an adjustment in a 10 pt document may not be correct if you change the document to 11 pt. If you work in multiples of the length `\baselineskip` then no extra adjustment is needed if the point size of the document is changed.

Because of the complicated way in which the row-spacing in an array is adjusted to fit the contents, the length given in `[ ]` is not exactly an *addition*. You have to experiment to find out what works best: doubling the length given does not usually double the space between the rows.

The existence of this optional length parameter to `\\` implies that no row of an array may start with `[`. If you need to do this, precede it with an emtpy pair of braces: `{}[....`

Changing the inter-row spacing for the whole array is much simpler than changing it for a single row. This spacing is governed by the value of the number `\arraystretch`, whose default setting is 1. Between the rows of an array, the length `\baselineskip` is multiplied by `\arraystretch`. In Figure 2, the rows have been spread out by putting

```
\renewcommand{\arraystretch}{2}
```

before the `\begin{array}`. However, this command was given *after* the start of the displayed Maths, so that it remained local to that particular display. If you experiment with other values of `\arraystretch` in this example you will find that there is virtually no separation between the first two rows when `\arraystretch` is smaller than 2. Figure 3 also shows an array in which `\arraystretch` has been altered.

Arrays inside arrays can have different values of `\arraystretch`. For example, in

```
\[
```

W_{1_G}	V_0	1	one stratum
W_ψ	$(V_{\text{rows}}+V_{\text{cols}})\cap V_0^\perp$	$2(q^2+q)$	direct sum of two unidentifiable strata
W_η	$(V_{\text{rows}}+V_{\text{cols}})^\perp$	q^3	one stratum

Figure 3. A table of mathematical information

$$\begin{aligned}
\alpha_1\colon p+q\omega+r\omega^2+s\omega^3+t\omega^4 &\mapsto p+q\omega+r\omega^2+s\omega^3+t\omega^4\\
\alpha_2\colon p+q\omega+r\omega^2+s\omega^3+t\omega^4 &\mapsto p+s\omega+q\omega^2+t\omega^3+r\omega^4\\
\alpha_3\colon p+q\omega+r\omega^2+s\omega^3+t\omega^4 &\mapsto p+r\omega+t\omega^2+q\omega^3+s\omega^4\\
\alpha_4\colon p+q\omega+r\omega^2+s\omega^3+t\omega^4 &\mapsto p+t\omega+s\omega^2+r\omega^3+q\omega^4
\end{aligned}$$

Figure 4. Four automorphisms of a splitting field

```
\renewcommand{\arraystretch}{2}
\begin{array}{ccc}
A & B &
   \renewcommand{\arraystretch}{1.2}
   \begin{array}{lr}
   ...
```

the outer array has `\arraystretch` equal to 2, while the inner array which is the third entry in the first row has `\arraystretch` equal to 1.2.

8.5 Changing the column spacing

The default inter-column spacing in an array is one quad. To change the spacing between a pair of adjacent columns, use `@{...}` between their alignment codes in the columns specifier, putting inside the braces the actual horizontal space required. For example, `\begin{array}{cc@{\qquad}c@{}c}` produces an array with four centred columns in which the inter-column spaces are, in order, one quad, two quads and zero.

At the start and end of the array there is usually a space of width half the inter-column space. These can be suppressed by putting `@{}` at the start and end of the columns specifier.

The overall inter-column spacing is controlled by the value of the length `\arraycolsep`, which is half the default inter-column space. This can be changed by using `\addtolength` in the usual way. I like to alter `\arraycolsep` by multiples of itself. For example, to treble the size of `\arraycolsep` do

```
\addtolength{\arraycolsep}{2\arraycolsep}
```

while to halve it make the last parameter `-0.5\arraycolsep`. In Figure 3, `\arraycolsep` has been multiplied by 2.5.

Finally, the command `\extracolsep` inside an `@{...}` can be used to insert extra space to the left of all subsequent columns: for example, `@{\extracolsep{1em}}`. This is not suppressed by a subsequent `@{...}`.

My remarks about quads above are not strictly true unless your current text font is Computer Modern roman in 10 pt. A quad length is defined to be 1 em, whose size depends on the current font, whereas `\arraycolsep` is defined to be 5 pt, irrespective of the point size of the document. I find it very odd that LaTeX, which has been so carefully set up so that the user can apparently change painlessly between 10 pt, 11 pt and 12 pt, does not change the size of Maths lengths like `\jot` and `\arraycolsep` in the different point sizes.

8.6 Words in arrays

The odd word or two in an array can be put in by using `\mbox`. But what should you do when you have a verbal phrase extending over more than one line, as in the final column of Figure 3?

Most people's first thought is to have a separate line of the array for each line of text. As usual, this visual approach is inferior to the logical one when it comes to revising your file. Hard experience has shown me that you have to

$$3(\mathbf{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu) = (x-1, y+2)\begin{bmatrix} 4 & 1 \\ 1 & 1 \end{bmatrix}\begin{pmatrix} x-1 \\ y+2 \end{pmatrix}$$

Figure 5. Matrices aligned on their top rows

keep altering which words go on which lines as you alter other parts of the array. It is better to use the fourth type of column code `p`, which creates a paragraph box of a specified width. Thus the columns specifier in Figure 3 is actually `cccp{1.5in}`. Just guess a length to put inside `p{...}` at first: changing it later is much simpler than shuffling words around.

Figure 3 demonstrates another good reason to use the `p` code. The rows are far apart, because `\arraystretch` has been magnified. But we *don't* want the lines of text at the end of the second row to be that far apart. We could make an inner array with a smaller value of `\arraystretch`, but it's simpler to use `p`.

The only problem with the naked `p` columns is that they are set justified on both sides, which is absurd in narrow columns. I usually put `\centering` or `\raggedright` at the beginning of every entry in such a column. Unfortunately, you cannot do this in the final column of an array, because LaTeX gets confused about whether `\\` refers to the row of the array or the line in the paragraph. If necessary, I get round this by adding a dummy unused column at the end of the array. Thus I told a lie above: the columns specifier in Figure 3 is really `cccp{1.5in}c`.

8.7 Inter-column constants

The code `@{...}` can be used for putting anything between two columns that must appear in every row, not just spaces. This is convenient for any repeated items. For example, the columns specifier in Figure 4 is

```
{l@{\colon p + q\omega + r \omega^2 +
s\omega^3 + t\omega^4 \mapsto p +
{}}r@{\omega + {}}r@{\omega^2 +
{}}r@{\omega^3 + {}}r@{\omega^4}}
```

while the first row is just

```
\alpha_1 &q & r & s & t
```

Notice the pairs `{}` that have been inserted to make $+$ behave as a binary operator.

8.8 Positioning

By default, each array is positioned so that it is centred vertically. In Figure 1, the equals sign is aligned with the centre of each of the three arrays. To align an array on its top row, start it with `\begin{array}[t]{...}`; to align it on its bottom row, use `[b]` in place of `[t]`.

These positionings work well for simple tables of numbers. However, the logical interaction between them and `\left` and `\right` fences produces bizarre results (try it!). If you want matrices aligned on their top rows, as in Figure 5, you have to do something quite complicated, which I shall come back to later.

8.9 Cases

We often write equations where the value on the right-hand side depends on some condition, as in

$$X(\omega,t) = \begin{cases} 1 & \text{if } \phi(\omega) = t \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

You can do the right-hand side of this using an `array`, but it is simpler to use the direct plain TeX command `\cases`. The right-hand side of Equation (1) was produced with

```
\cases{1 & if $\phi(\omega) = t$\cr
       0 & otherwise.}
```

The argument of `\cases` is like a two-column array, with its rows separated by `\cr` instead of `\\`. The first entry in each row is in Maths mode; the second is in non-Maths mode. Both columns are left-aligned.

8.10 Binomial coefficients

If you write your binomial coefficients as $\binom{6}{2}$ rather than ${}^6\mathrm{C}_2$ you may be tempted to build them with an `array`. Don't. Use the plain TeX infix command `\choose`, as follows.

`6 \choose 2` $\qquad \binom{6}{2}$

It comes out rather differently in `\textstyle` and in `\displaystyle`.

Because `\choose` does not have its two arguments in braces, it is safest if you enclose the whole expression in grouping braces, as in `{6 \choose 2}`, to ensure that `\choose` does not search too far back or too far forward for its arguments. If you need to use this a lot, make a LaTeX-style macro such as

```
\newcommand{\binom}[2]{{#1\choose#2}}
```

The command `\atop` works similarly, but omits the parentheses.

Exercises

Exercise 55 The matrix $\begin{bmatrix} 1 & 3 \\ 3 & 5 \end{bmatrix}$ is symmetric.

Exercise 56 The determinant is given by

$$\det A = \begin{vmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{vmatrix}$$

Exercise 57 For a rotation,

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x\cos\theta + y\sin\theta \\ y\cos\theta - x\sin\theta \end{pmatrix}.$$

Exercise 58 Here is a pair of mutually orthogonal Latin squares:

$$\begin{array}{ccc} A & B & C \\ B & C & A \\ C & A & B \end{array} \qquad \begin{array}{ccc} \alpha & \beta & \gamma \\ \gamma & \alpha & \beta \\ \beta & \gamma & \alpha \end{array}$$

Square 1 Square 2

Exercise 59 A few real functions:

$x \mapsto x^2$	differentiable everywhere
$x \mapsto \lvert x \rvert$	differentiable everywhere except at the origin
$x \mapsto \lfloor x \rfloor$	continuous everywhere except at the integers

Exercise 60 *Typeset the table in Figure 3.*

Exercise 61 Pascal's Triangle begins

$$\begin{array}{ccccccccccccc} &&&&&& 1 &&&&&& \\ &&&&& 1 && 1 &&&&& \\ &&&& 1 && 2 && 1 &&&& \\ &&& 1 && 3 && 3 && 1 &&& \\ && 1 && 4 && 6 && 4 && 1 && \\ & 1 && 5 && 10 && 10 && 5 && 1 & \\ 1 && 6 && 15 && 20 && 15 && 6 && 1 \end{array}$$

Exercise 62 The Stirling numbers of the second kind are defined by

$$S(n,k) = \frac{1}{k!} \sum_{j=1}^{k} (-1)^{k-j} \binom{k}{j} j^n.$$

Year 1	Year 2	Year 1	Year 2	Year 1	Year 2	Year 1	Year 2
269	466	134	590	211	505	209	352
100	219	74	137	102	363	29	254
197	421	88	356	191	563	19	106
216	708	25	212	67	338	44	268

Figure 6. Number of eelworms in two successive years

Exercise 63 If the random variable X is symmetric about 0 and if $Y = X^2$ then

$$f_Y(y) = \begin{cases} 0 & \text{if } y \le 0 \\ \dfrac{f_X(\sqrt{y})}{\sqrt{y}} & \text{if } y > 0. \end{cases}$$

Exercise 64

$$\begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n} \\ \vdots & \cdots & \ddots & \vdots \\ \sigma_{1n} & \cdots & \cdots & \sigma_{nn} \end{bmatrix}$$

Exercise 65

$$\begin{array}{ccccccccc} M_0 & \subseteq & M_1 & \subseteq & M_2 & \subseteq & \cdots & \subseteq & M_m \\ \uparrow & & \uparrow & & \uparrow & & & & \\ \text{null} & & \text{linear} & & \text{quadratic} & & & & \end{array}$$

9 Tables of data

9.1 General

The extra topics that I cover in this section are most pertinent to tables of data, but do sometimes arise in Mathematical arrays. Likewise, many considerations about Mathematical arrays also apply to tables of data. For example, the table in Figure 6 does not really have eight columns: it has two long columns but has been broken down into four parts to fit on the page nicely. So it is set as an array with one row and four columns, each of whose entries is an array with two columns.

LaTeX has another environment, called `tabular`, which is very similar to `array`. It may be used inside or outside Maths mode. By default, each of its entries is set in ordinary mode. However, any extra space on either side of the ampersands has no effect. The length `\arraycolsep` is replaced by the length `\tabcolsep`. Tables of data often have a mix of words and symbols, and you have to decide which is less trouble: using an `array` and putting the words in `mbox`es or `p` columns, or using a `tabular` and putting the symbols in `$...$`. If the latter, don't forget to put numbers in `$...$` if they involve a minus sign.

9.2 Horizontal rules

To obtain a horizontal rule stretching the full width of the array, use `\hrule`. Don't forget to finish the previous row with `\\`, even if it is the last row of the array, as at the end of Figure 7. Two or more `\hrule`s can follow each other with no intervening `\\`. A little vertical space is put between two such horizontal rules. However, any vertical rules in the array (see below) will be suppressed between the successive horizontal rules.

For a horizontal rule spanning only columns 2–4, say, use `\cline{2-4}`. Again, finish the preceding row with `\\`. Figure 7 shows an example of this. Successive `\cline`s will be interpreted as being in the same row, as in `\cline{2-4}\cline{7-7}`, for example, so that `\cline{2-4}\cline{2-4}` does *not* produce two horizontal lines with a little space between them. Likewise, there is no space between an `\hline` and an immediately preceding or succeeding `\cline`.

	1961	1971	1981
Billion passenger kilometres travelled by:			
Air	1	2	3
Rail	39	36	34
Road			
Buses and coaches	67	51	42
Cars, taxis and two-wheeled motor vehicles	142	284	380
Bicycles	10	4	4
Total	259	377	406

Figure 7. Passenger transport (from [1])

Premiums	
1 Month	£ 18
3 Months	£ 48
6 Months	£ 95
9 Months	£136

Figure 8. Travel insurance

Bootstrap	0.301
Jackknife	0.314
Delta method	0.244*
Half-samples	0.364†
Random subsampling	0.423‡
Normal theory	0.302

Figure 9. Sticking out to the right (from [2])

9.3 Vertical rules

A vertical rule is obtained by putting `|` at the appropriate place in the columns specifier. For example, the columns specifier for the `tabular` in Figure 8 is

```
|r@{ Month}l@{\quad\pounds}r|
```

Two or more `|`s may come next to each other.

For a vertical rule in a single item, stretching from the top to bottom of its row, use `\vline`. This cannot be used in a `p` column, but may be put inside `@{...}`.

If you omit the trailing ampersands in a row with few items the trailing vertical rules will also be omitted. This can be a nuisance in a sparse table. If you want a vertical rule at each side of a sparse table, consider doing this:

```
\begin{array}{|c|}
\begin{array}{@{}...@{}}
...
\end{array}
\end{array}
```

The length `\arrayrulewidth` controls the thickness of all the horizontal and vertical rules just described. The width of the space between the vertical rules created by `||` in a the columns specifier is controlled by the length `\doublerulesep`, as is the vertical space between two successive `\hline`s.

9.4 Spanning several columns

The `\multicolumn` command is used to create an entry spanning several columns. It takes three arguments. The first is the number of columns to be spanned; the second is the code for the type of column to be used; the third is the text of the entry.

```
\begin{tabular}{crrrr}
Type & \multicolumn{4}{c}{Grade}\\
\cline{2-5}
 & 12 & 16 & 24 & 30\\
\hline
$A$ & 0 & 1 & 21 & 24\\
```

```
$B$ & 1 & 6 & 24 & 13
\end{tabular}
```

Type	Grade			
	12	16	24	30
A	0	1	21	24
B	1	6	24	13

Any `|` in the columns specifier is regarded as belonging to its preceding column, and may be overwritten by `\multicolumn` unless it is included in the second argument. The exception is any `|` to the left of the first column, which is regarded as part of the first column. For example, the first row in Figure 8 is

```
\multicolumn{3}{|c|}{Premiums}\\
```

Expressions `@{...}` in the columns specifier are similarly assigned to columns and similarly overwritten.

Indented row labels for subheadings can also be conveniently obtained by using `\multicolumn`. The `tabular` in Figure 7 begins with three `l` columns. The first row after the second horizontal rule is a `\multicolumn` spanning three columns; the next begins with & followed by a `\multicolumn` spanning two columns.

Unfortunately, if the width of a multicolumn entry spanning three or more columns is more than the total width of the spanned columns, LaTeX cannot adjust the spacing between those columns to make it equal: it puts all the extra space to the left of the last of the spanned columns. The only way that I know around this is to use `@{...}` to put the same amount of extra space between each adjacent pair of the spanned columns.

9.5 *Changing the type of an entry*

Although its name does not suggest it, the command `\multicolumn` is also used to change the type of a single entry. This is most often used to give a centred heading to a column of right-aligned data.

```
\begin{array}{rrr}
\multicolumn{1}{c}{n} &
\multicolumn{1}{c}{n^2} &
\multicolumn{1}{c}{\phi(n)}\\
9 & 81 & 6\\
10 & 100 & 4\\
11 & 121 & 10
\end{array}
```

$$\begin{array}{rrr}
\multicolumn{1}{c}{n} & \multicolumn{1}{c}{n^2} & \multicolumn{1}{c}{\phi(n)}\\
9 & 81 & 6\\
10 & 100 & 4\\
11 & 121 & 10
\end{array}$$

As this example shows, this works fine when the centred heading is narrower than the widest other entry in the column. However, when it is wider then all the other entries in the column are aligned with its right-hand edge, when probably what you wanted was to have them right-aligned among themselves and then all centred under the heading. How can this be fixed?

If most of the numbers are the same length, a reasonable solution is to make the whole column centred and put in `\phantom{0}`s to make the shorter numbers as long as the longest. This works because all digits have the same width. That is what I did in Figure 6.

I adopted a different solution in Figure 7, where I decided that the year headings, not being numbers to be added to other numbers in their columns, should be centred. This time I used the columns specifier `rrr` for each column of data. In the headings I used `\multicolumn{3}{c}{...}` and in the data rows I left one blank entry either side of each data entry. The result is rather widely spaced: space could be saved by using `@{...}` between each data column and its blank neighbours.

The command `\multicolumn{1}` can also be used to remove a vertical rule or the contents of an `@{...}` from one particular row. For example, if the columns specifier is `l|r` then the vertical rule is missing from any row whose first entry is made with `\multicolumn{1}{l}`.

9.6 Decimal tabulation

Sometimes a column of numbers with decimal points should be aligned on those decimal points. If they all have the same number of digits after the point, there is no problem: simply use an `r` column. If only a few numbers are missing a few digits after the point, fill in the spaces with `\phantom{0}` and still use an `r` column. Only if you have a long column of numbers with very variable numbers of digits after the point is it worth going to the trouble of using the columns specifier `r@{.}l` and entering a number such as 3.142 as `3 & 142`.

9.7 Things that stick out

Sometimes you have a column of numbers that should be right-aligned but some of the entries have accretions that should stick out to the right, as in Figure 9, or a left-aligned column with pieces sticking out to the left. There are three ways of dealing with this, each appropriate in different circumstances.

If the pieces that stick out are all digits, use `\phantom{0}` as described above.

Sometimes the piece that sticks out is very small and occurs in only one row. If a displayed Mathematical array ends the sentence then sometimes the full stop needs to come outside the array, vertically centred (with `\end{array}.\]`); but sometimes the final row of the array needs a full stop at the end of it without upsetting the column alignment. Then the trick is to add the full stop in such a way that (La)TeX thinks that it doesn't take up any space. According to [3, page 98], you do this with `\makebox[0pt][l]{.}`, but the plain TeX `\rlap{.}` is shorter and achieves the same effect. For a small item sticking out on the left use `\makebox[0pt][r]{ }` or `\llap{ }`. Of course, the content of `\makebox{ }` or `\rlap` or `\llap` is not in Maths mode.

Statisticians sometimes put a question mark to indicate dubious data, and scientists often decorate numbers with stars to show statistical significance. In such cases it is best to use the columns specifier `r@{}l` for the column: the `r` part is used for the numbers and the `l` part is used for any decorations. In Figure 9 the columns specifier is `lr@{}l` and the fourth line is

```
Half-samples & 0.364 & \dag\\
```

To see the difference between these last two solutions, compare the following.

```
\begin{array}{|r@{}l|}
20 & ++
\end{array}
```

$\begin{array}{|r@{}l|} 20 & ++ \end{array}$

```
\begin{array}{|r|}
20 \rlap{$++$}
\end{array}
```

$\begin{array}{|r|} 20 \rlap{$++$} \end{array}$

9.8 Columns specifiers

We have seen that a columns specifier is a string of the following objects: `r`, `l`, `c`, `p{...}`, `@{...}` and `|`. It is easy to lose track when setting up the column specifier for a complicated array. However, repeated elements may be simplified by using `*`. Thus an array with 15 centred columns can be specified with `*{15}{c}`. This shorthand is really advantageous for something complicated like `l|*{4}{r@{}lrr|}`. Moreover, `*` expressions can be nested within each other.

Exercises

Exercise 66

Values of $\nu_{\gamma\chi}$ for the design

γ	χ_0	χ_1	χ_2	χ_3	χ_4
μ	1	0	0	0	0
σ	1	0	0	0	0
ρ	1	0	0	$\frac{1}{4}$	0
κ	1	$\dfrac{3+e_2-e_1}{36}$	$\dfrac{3+e_4-e_2}{36}$	0	$\dfrac{3+e_1-e_4}{36}$
ε	1	1	1	1	1

Exercise 67 Table of results:

	B_1	B_2	
A_1	37	53	90
A_2	41	48	89
	78	101	179

Exercise 68 *Typeset the table in Figure 6.*

Exercise 69

Source	df	SS	MS	VR
x_1	1	104474	104474	1220.5
$x_2 \mid x_1$	1	2284	2284	26.68
regression	2	106758		
residual	15	1284	85.6	
‘total’	17	108042		

Exercise 70 *Typeset the table in Figure 7.*

Exercise 71

Date	Miles	Gallons	Cost
27 December	46674	4.18	£ 5.56
3 January	46038	3.93	£ 5.00
6 January	47956?	7.1	£10.50
10 January	47292	7.89	£10.50
17 January	47464	4	£ 5.00
....			
3 February	48112	6.88	£ 9.00

References

[1] CENTRAL STATISTICAL OFFICE: *Key Data 1986*, Her Majesty’s Stationery Office, London, (1986).

[2] EFRON, B.: *The Jackknife, the Bootstrap and Other Resampling Plans*, Society for Industrial and Applied Mathematics, Philadelphia, (1982).

[3] LAMPORT, L.: *LaTeX: A Document Preparation System*, first edition, Addison-Wesley, Reading, Mass., (1986).

[4] LAMPORT, L.: *LaTeX: A Document Preparation System*, second edition, Addison-Wesley, Reading, Mass., (1994).

IV Maths in LaTeX: Part 6, Harder arrays

R. A. Bailey
Queen Mary and Westfield College,
University of London

1 Recall

This is the sixth and final part of a sequence of tutorials on typesetting Mathematics in LaTeX. The first five appeared in issues 4.4, 4.5, 5.1, 5.2 and 5.3 of *Baskerville*. The series includes some things which can be found in [3], but I am working in more things which, while straightforward and necessary for Mathematical work, are not in [3] or [4]. In this final tutorial I cover the harder parts of arrays, including aligned equations.

In case you missed the previous tutorials, I remind you that I expect you, the reader, to do some work. Every so often comes a group of exercises, which you are supposed to do. Use LaTeX to typeset everything in the exercise except sentences in italics, which are instructions. If you are not satisfied that you can do the exercise, then tell me. Either write to me at

School of Mathematical Sciences
Queen Mary and Westfield College
Mile End Road
London E1 4NS

with hard copy of your input and output, or email me at `r.a.bailey@qmw.ac.uk` with a copy of the smallest possible piece of LaTeX input file that contains your attempt at the answer. In either case I will include a solution in the following issue of *Baskerville*: you will remain anonymous if you wish.

This tutorial covers things that LaTeX is not really very good at. You may ask why I have not simply referred you to the packages `amsmath`, `array` and `delarray`. One reason is that every package has its price: it may give you the functionality that you want at the expense of changing something that you are quite happy with. The other is that you often are not allowed to include style packages when you submit an article to a journal or conference proceedings. It is your choice whether to use the inelegant solutions presented here or to cut out the relevant pieces of code from various style packages.

2 Answers

2.1 Boxed subarrays

In the panel session at the end of the UKTUG meeting on ‘The New Maths for the New LaTeX’ on 7 June, one of the participants asked how to create an array in which there is a box around a subarray, as in

$$\begin{array}{c|cccc|}
\multicolumn{1}{c}{1} & 2 & 3 & 4 & \multicolumn{1}{c}{5}\\
\cline{2-5}
2 & A & B & C & D\\
3 & B & A & D & C\\
4 & C & D & A & B\\
5 & D & C & B & A\\
\cline{2-5}
\end{array}$$

The answer is to use `\cline` for the horizontal sides of the box and to put `|` in the columns specifier to obtain the vertical sides of the box, overriding this with `\multicolumn{1}` where necessary. Thus the input for the preceding array begins

```
\begin{array}{c|cccc|}
\multicolumn{1}{c}{1} & 2 & 3 & 4 &
\multicolumn{1}{c}{5}\\
\cline{2-5}
2 & A & B & C & D\\
```

```
...
```

2.2 Angle brackets

Several people have asked me why I insist that `\langle` and `\rangle` should be used for angle brackets when they prefer the shape of < and >. At a group theory conference in July I saw a good, if unconscious, demonstration of why < and > should not be used. A line of displayed Maths on an overhead projector transparency was

$$G =< a,b,c > \times < a,c,e >^x$$

Look at the spacing. TeX knows that $=$ and $<$ are both relations, so it puts no space between them, but it does put some space between the relation $>$ and the binary operator $\times$. If you put this equation in the running text, you will find that the line may break between the $<$ and the a. If you really prefer the shapes of $<$ and $>$ to $\langle$ and $\rangle$ then you should make yourself macros such as

```
\newcommand{\llangle}{\mathopen{<}}
\newcommand{\rrangle}{\mathclose{>}}
```

Then the display becomes

$$G = \mathopen{<}a,b,c\mathclose{>} \times \mathopen{<}a,c,e\mathclose{>}^x$$

However, you cannot make these new angle brackets expand by preceding them with `\left` and `\right`.

12 Arrays of equations

12.1 Don't do it

Many of us write our lecture notes on the board as a series of equations, more or less aligned, and are tempted to write in print in the same fashion. Don't. For one thing, printed material needs the connecting words that you normally say at the board, such as 'and' or 'it follows that' or 'substituting for … '. For another, alignment suggests to the reader that the equations are somehow related, so it should not be used merely because two displayed equations come one after another with no intervening text: use two separate lines of displayed Maths instead, using `\[` and `\]`.

12.2 Parallel definitions

For two or more parallel or analogous definitions or results, use the `eqnarray*` environment. If a typical line is $A = B$ then type that line as `A & = & B` and put `\\` at the end of each line except the last. Extra space can be added after any `\\` just as with ordinary arrays. For example, the parallel definitions of $\cap$ and $+$

$$\begin{aligned} W \cap X &= \{v \in V : v \in W \text{ and } v \in X\} \\ W + X &= \{w + x : w \in W \text{ and } x \in X\} \end{aligned}$$

have input

```
\begin{eqnarray*}
W \cap X & =& \left\{v\in V:v \in W
\mbox{ and } v\in X\right\}\\
W + X   & = & \left\{w+x:w\in W
\mbox{ and } x\in X\right\}
\end{eqnarray*}
```

12.3 Chains of equalities

The `eqnarray*` environment is also useful for a chain of equalities or inequalities, such as

$$\begin{aligned} \sum_{i=1}^{q} x_{ij}(M - x_{ij}) &= M^2 - \sum_{i=1}^{q} x_{ij}^2 \\ &\leq M^2 - \frac{M^2}{q} \\ &= \theta M^2. \end{aligned}$$

Here each line after the first begins with & followed by = or some other relation, followed by another &. Any line may conclude with `\qquad\mbox{...}` to give a short explanation, just as in a single line of displayed Maths.

$$\begin{aligned} n(Q_3-Q_1)^4\,\mathrm{Var}\,W &= \left[(M-Q_1)\left(\frac{1}{f_M}-\frac{1}{2\,f_{Q_3}}\right)+(Q_3-M)\left(\frac{1}{f_M}-\frac{1}{2\,f_{Q_1}}\right)\right]^2 \\ &\quad +\frac{1}{2}\left[\left(\frac{M-Q_1}{f_{Q_3}}\right)^2+\left(\frac{Q_3-M}{f_{Q_1}}\right)^2\right] \end{aligned}$$

Figure 1. An overlong equation

12.4 Overlong displays

Sometimes what is conceptually a single line of displayed Maths, whether it is an equation or not, is simply too long to fit on one line. Then you can use `eqnarray*`, choosing where to split the line. If you split it at a binary operator, it is usual to put the binary operator after the split. In this case you must precede it with `\mbox{}` so that TeX knows that it is a binary operator. The two lines in Figure 1 are given by

```
...\right]^2\\ & & \mbox{} + \frac{1}{2} ...
```

To split an even longer line, you may want the second and succeeding lines to come partly underneath the first line. You can do this by enclosing the whole of the first line in `\lefteqn{ }`, thus fooling TeX into thinking that it has no width. Starting subsequent lines with `& &` gives that necessary bit of indentation. In this example

$$\begin{aligned} &\sum\left\{\sum\{f(B):B\cap A=\emptyset\}:A\supseteq J\right\} = \\ &\qquad \sum\left\{\sum\{f(B):A\supseteq J,\ A\cap B=\emptyset\}:B\cap J=\emptyset\right\} \end{aligned}$$

the lines begin

```
\lefteqn{\sum ...
 & & \sum ...
```

12.5 Numbered aligned equations

The environment `eqnarray` works just like `eqnarray*` except that each line is numbered, in the same sequence as `equation`s. If you want any line to be not numbered, just put `\nonumber` before the end of the line. If you want to refer somewhere else to the number, put a `\label` on the line in the usual way. Thus

$$\begin{aligned} \bar{F}(x_1,x_2) &= \int_0^\infty \exp(-\theta a_1 x_1^c - \theta a_2 x_2^c)\frac{\theta^{b-1}\lambda^b e^{-\theta\lambda}}{\Gamma(b)}\,d\theta \\ &= \frac{\lambda^b}{(\lambda+a_1x_1^c+a_2x_2^c)^b} \qquad (1) \end{aligned}$$

is created with

```
\begin{eqnarray}
\bar{F}(x_1,x_2) & = & \int_0^\infty ...
{\rm d}\theta \nonumber \\
 & = & \frac{ ...
\end{eqnarray}
```

12.6 What is `eqnarray`*?*

The two environments `eqnarray` and `eqnarray*` differ only in the numbering of lines. Each creates a piece of displayed Maths containing a special sort of array. The array has only three columns. The first column is in `\displaystyle` and is right-aligned. The second is in `\textstyle` and is centred. The third is in `\displaystyle` and is left-aligned. The space between columns is controlled by `\arraycolsep` just as for ordinary arrays. The space between rows is (unless you put something after the `\\`) what you would get in an ordinary array by putting `\\[\jot]`.

12.7 Simultaneous equations

Simultaneous equations are often written with a vertical alignment for each variable and for the binary operators in between them, as well as for the equals sign, as the following example shows.

$$\begin{array}{rcrcrcrcrl}
x_1 & - & x_2 & + & x_3 & - & x_4 & + & x_5 & = 1 \\
2x_1 & - & x_2 & + & 3x_3 & & & + & 4x_5 & = 2 \\
3x_1 & - & 2x_2 & + & 2x_3 & + & x_4 & + & x_5 & = 1 \\
x_1 & & & + & x_3 & + & 2x_4 & + & x_5 & = 0
\end{array}$$

This is too many alignments for an `eqnarray*`, so an `array` has been used in displayed Maths, with every line ending with `\\[\jot]`. With a column for each variable and one for each binary operator, almost all pairs of adjacent columns should have the separation that TEX normally gives between an ordinary Maths symbol and a binary operator, which is `\medmuskip`. Unfortunately, you cannot set `\arraycolsep` to be equal to `0.5\medmuskip`: TEX will complain. So I have set `\arraycolsep` to zero: the command

```
\setlength{\arraycolsep}{0pt}
```

has been placed before the array but within the displayed Maths, to limit its scope. Then the columns specifier `{*{4}{rc}r@{{}={}}r}` does the trick for the equals sign, which comes in every column: for the binary operators I have put `{}+{}` or `{}-{}` as least once in each column.

There are two other possibilities that could be used here. The `array` package allows you to put items in the columns specifier that will be incorporated in array entries before the boxes are made. So you could put the `{}` either side of each binary operator by putting it in the columns specifier once and for all. See [2, Section 5.3]. That would be useful if the binary operators in the array had differing widths. The second is to effectively set `\arraycolsep` equal to `0.5\medmuskip`. Now, `\medmuskip` is 4 mu plus some stretchability, and 1 mu is 1/18 of an em in the current font. So you can do

```
\setlength{\arraycolsep}{0.11em}
```

and omit all the `{}`, so long as the current font does not change (by too much) between the issuing of that command and the setting of the entries in the array.

12.8 Which to use: `eqnarray` *or* `array`*?*

Regular readers will know that I am a big fan of LATEX. All the same, I think that the design of `eqnarray` is fundamentally flawed. It is not simply a method of aligning lines of displayed Maths, chiefly because it uses `\arraycolsep` to insert larger spaces than normal, but also because it changes between `\displaystyle` and `\textstyle` and because it is limited to three columns. For these last two reasons, it is also not a method of achieving a displayed array all of whose entries are in `\displaystyle` and whose rows are spread out, which would have been a useful environment.

So which should you use, `eqnarray` or `eqnarray*` or `array`? Each of them needs some work to give good results.

If you need a set of aligned equations carrying a single number then I recommend using `array` inside an `equation`. You will have to put in `\displaystyle` and `\jot` where necessary. If one or more lines must be individually numbered then there is nothing for it but to use `eqnarray`.

If an unnumbered set of aligned equations has only two alignment points you may be able to use `eqnarray*` if you are careful about the inter-column spacing. Thus if you put an `&` on only one side of an equals sign you must put a quad space on the other side. In the following display each line has the form

```
... \quad = & ... & \qquad ...
```

$$\begin{array}{rcll}
g(x) & = & e^x & \qquad \text{for } x \in \mathbb{R}, \\
h(y) & = & \ln y & \qquad \text{for } y > 0, \\
h'(y) & = & \frac{1}{y} & \qquad \text{for } y > 0.
\end{array}$$

For a set with more alignment points, such as

$$\begin{array}{rlrlrlrl}
f(1) = & 1 & f(2) = & 0 & f(3) = & -2 & f(4) = & 3 \\
g(1) = & 5 & g(2) = & 7.5 & g(3) = & 6 & g(4) = & -4
\end{array}$$

or simultaneous equations, use `array` and be cunning with the columns specifier.

For parallel results, or for chains of (in)equalities, it would be good to have a form of `eqnarray` and `eqnarray*` in which the space on either side of the equals sign is what TeX normally puts between a relation and an ordinary Maths symbol, which is `\thickmuskip`. Now, `\thickmuskip` is 5 mu plus some stretchability, so we can use the same fudge that we used for simultaneous equations. It is no good changing `\arraycolsep` globally, because that would affect other `array`s. So you could make an environment to use in place of `eqnarray` such as the following.

```
\newenvironment{bettereqnarray}%
{\setlength{\arraycolsep}{0.14em}%
 \eqnarray}%
{\endeqnarray}
```

Compare the following display, made with `bettereqnarray` and `\nonumber`, with the previous form made with `eqnarray*`. Now that the spaces around the aligned = are correct, a second = can be placed on the same line.

$$\begin{array}{rcl}\sum_{i=1}^{q} x_{ij}(M-x_{ij}) &=& M^2-\sum_{i=1}^{q} x_{ij}^2 \\ &\leq& M^2-\frac{M^2}{q} = \theta M^2\end{array}$$

If you are uneasy about that fudge, set `\arraycolsep` to zero. Then put `& {}={} &` instead of `& = &` in the centre of the array.

There is a disadvantage common to both of these `bettereqnarray` environments: if you have any ordinary `array` within them then the value of `\arraycolsep` will almost certainly be wrong and you will have to reset it locally.

There are several better environments for aligned equations in the `amsmath` package, which is described in [1]. However, it does not seem to be possible to obtain these environments without the rest of the package, which you may not want: for example, it disables `\over`.

13 Exercises

Exercise 72 Möbius inversion gives:

$$B_\gamma = \sum_{\alpha\in\Gamma} z(\gamma,\alpha)S_\alpha, \tag{2}$$

$$S_\alpha = \sum_{\gamma\in\Gamma} m(\alpha,\gamma)B_\gamma. \tag{3}$$

Exercise 73 *Get the number cited here from the question above, by cross-reference.*

Now

$$\begin{array}{rcll} L_\alpha &=& X'S_\alpha X & \\ &=& \sum_{\gamma\in\Gamma} m(\alpha,\gamma)X'B_\gamma X & \qquad \text{by Equation (3)} \\ &=& \sum_{\gamma\in\Gamma} \frac{m(\alpha,\gamma)}{k_\gamma}C_\gamma. & \end{array}$$

Exercise 74

$$\begin{array}{rcl}\sum a_j b_k &=& a_1b_1+a_1b_2+a_1b_3 \\ && +a_2b_1+a_2b_2+a_2b_3 \\ && +a_3b_1+a_2b_2+a_3b_3. \end{array} \tag{4}$$

Exercise 75

$$\begin{array}{rcl}\sum_{1\le j,k\le 3} a_j b_k & = & a_1b_1+a_1b_2+a_1b_3 \\ & & +a_2b_1+a_2b_2+a_2b_3 \\ & & +a_3b_1+a_2b_2+a_3b_3.\end{array} \tag{5}$$

Exercise 76 Using the hint, we get

$$3(1-z)^2\sum_k \binom{1/2}{k}\left(\frac{8}{9}z\right)^k (1-z)^{2-k} =$$
$$3(1-z)^2\sum_k \binom{1/2}{k}\left(\frac{8}{9}\right)^k \sum_j \binom{k+j-3}{j} z^{j+k}$$

and now look at the coefficient of z^{3+l}.

Exercise 77 Solve the system of equations

$$\begin{aligned} 2x+\ \ y+5z&=4\\ 3x-2y+2z&=2\\ 5x-8y-4z&=1.\end{aligned}$$

Exercise 78 The dyad appears as

$$\begin{array}{rcl}\mathbf{AB} & = & A_xB_x\mathbf{ii}+A_xB_y\mathbf{ij}+A_xB_z\mathbf{ik} \\ & & +A_yB_x\mathbf{ji}+A_yB_y\mathbf{jj}+A_yB_z\mathbf{jk} \\ & & +A_zB_x\mathbf{ki}+A_zB_y\mathbf{kj}+A_zB_z\mathbf{kk}\end{array} \tag{6}$$

Exercise 79

$$\begin{array}{ll} f(x)=1 & f=(1,\ 1,\ 1,\ \ 1,\ \ 1) \\ g(x)=x & g=(1,\ 2,\ 3,\ \ 4,\ \ 5) \\ h(x)=x^2 & h=(1,\ 4,\ 9,\ 16,\ 25).\end{array}$$

14 Hand-crafting alignments

In this section I show a few tricks for difficult alignments. I show them because I know that I am not the only Mathematician who needs to produce these effects. I am not particularly proud of the methods I have used, because in each case I have had to either measure a length explicitly or use phantoms: I couldn't find a way of getting the right sizes automatically. So if any reader can write in with a better way of doing these things, I shall be delighted.

14.1 Horizontal braces in arrays

When I use an `array` to show a chain of equalities, I often use an under- or overbrace to indicate how terms are grouped. Here is an example.

$$\begin{array}{rcccccc} y & = & \multicolumn{3}{c}{f} & + & r \\ & = & \overbrace{\bar{y}(1,\ldots,1) & + & \dfrac{\mathrm{CS}(x,y)}{\mathrm{CS}(x,x)}x'} & + & r \\ & & \uparrow & & \uparrow & & \uparrow \\ & & \text{fit for null} & & \text{extra fit} & & \text{residual} \\ & & \text{model} & & \text{for straight} & & \\ & & & & \text{line} & & \\ & & & & \text{model} & & \end{array}$$

$$3(\mathbf{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu) = (x-1, y+2)\left[\begin{array}{rr} 4 & 1 \\ 1 & 1 \end{array}\right]\left(\begin{array}{c} x-1 \\ y+2 \end{array}\right)$$

Figure 2. Matrices aligned on their top rows

The problem is that the brace needs to span several columns. So the brace needs to be put in with a `\multicolumn` command, so it cannot automatically be set to the correct width. I solve this by using `\hphantom` to obtain the width of the spanned columns. Here there is an entry

```
\multicolumn{3}{c}{f}
```

in the first row; and the corresponding entry in the second row is

```
\multicolumn{3}{c}{
\overbrace{\hphantom{\bar{y} (1,\ldots, 1)
\quad+\quad ... x’}}}
```

I have used the fact that the intercolumn space is one quad.

An alternative solution is to use the command `\downbracefill` as the final argument of the `\multicolumn` in the second row. However, you have to be outside Maths mode to use `\downbracefill`, and putting it inside an `mbox` is no good because that destroys the expandability. So you have to set the whole thing in a `tabular` environment rather than `array`, and then enclose every other entry in `$ $`, which is a pain. There is also an analogous command `\upbracefill`.

14.2 Paragraph boxes in arrays

In the preceding display I have also set some explanatory text in paragraph boxes whose width is determined by the width of a Mathematical expression in the same column. For each such column I make a new length and use `\settowidth` to make it as wide as the desired Mathematical expression: see [3, page 95] or [4, page 101]. Then I make a `p` column element of that width.

For example, in the first column I get the correct width with

```
\newlength{\gnat}
\settowidth{\gnat}{$\bar{y} (1,\ldots, 1)$}
```

and then use it as follows.

```
\multicolumn{1}{p{\gnat}}{\centering
fit for null model}
```

14.3 Top-aligned matrices

In the last article I showed Figure 2 as an example of something that is not easy to do in LaTeX. To achieve this I have made a macro `\topthing` which takes as its single argument any item that must be aligned with the top rows of the arrays. The biggest arrays here have two rows, so `\topthing` produces a two-rowed array whose second row is empty. If there were bigger arrays here I would have to have a macro for each smaller size of array. The empty second row of the array contains a phantom zero: this works because zero is the standard height, as are all the entries that occur in the second rows of arrays. I don’t know how to fudge this for arbitrary heights of entries. Finally, I remove the extra space that is usually put on each side of an array.

The `\topthing` macro is defined by

```
\newcommand{\topthing}[1]%
  {\begin{array}{@{}c@{}}
   #1\\\phantom{0}
   \end{array}}
```

Then the equation in Figure 2 is done with

```
\topthing{3(...   (x-1, y+2)}
\left[\begin{array}{rr}
4&1\\1&1
\end{array}
...
```

Top-aligned matrices can be done very simply if you have access to the package `delarray`. Get it. See [2, Section 5.3.6]. However, be warned that `delarray` inputs the package `array`, which makes a small difference to the way that `|` works in all `array`s.

14.4 Bordered matrices

Using `\left[` and so on to get the right size of fences around a matrix works fine for unbordered matrices. But how do you get a bordered matrix such as the following?

$$\begin{array}{cc} & \begin{array}{ccccc} \emptyset & \{1\} & \{2\} & \{1,2\} & \{1,2,3\} \end{array} \\ \begin{array}{c} \emptyset \\ \{1\} \\ \{2\} \\ \{1,2\} \\ \{1,2,3\} \end{array} & \left[\begin{array}{ccccc} 1 & -1 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right] \end{array} \tag{7}$$

The command `\bordermatrix` given by plain TEX will not do, because it uses parentheses for the fences and does not let you choose how the columns are aligned.

The problem is to get the fences of the right size while keeping the alignment of the two borders with the body of the matrix. My solution is to set the whole thing as a 2×2 array with no space between the columns. The top left entry is empty. The top right entry is a one-rowed array containing the column labels. The bottom left entry is a one-columned array containing the row labels. The bottom right entry is the body of the matrix, including the fences.

If all entries have the same width and height, that's enough. However, if any column in the body of the matrix has a different width from the corresponding column in the top border, then the narrower of the two has to be expanded. Set a new length equal to the wider item and then put the smaller item in a `makebox` of that width.

In the matrix (7) it is clear that, in every column, it is the label that is the widest element. So I made a macro to set the first row of the body of the matrix to the width of the labels, as follows.

```
\newlength{\perch}
\newcommand{\fish}[2]%
{\settowidth{\perch}{$#1$}
 \makebox[\perch]{$#2$}}
```

For example, the first entry in the main body of the matrix is

```
\fish{\emptyset}{1}
```

Note that it suffices to make a single entry in each column of the body as wide as the column label.

(By the way, I can never decide whether the minus signs should be taken into account or not when centering the columns of such a matrix. Here I decided not to, and so I defined

```
\newcommand{\minone}{\llap{$-$}1}
```

and then used `\minone` for each -1.)

If the row heights in the body of the matrix do not match those in the array of row labels, as in the following matrix, they can be adjusted with `\vphantom`. If `X` is the tallest item in the row, simply put `\vphantom{X}` in a single entry of the corresponding row of the other array: it will not affect the horizontal spacing.

$$\begin{array}{cc} & \begin{array}{cccccc} 1 & \ldots & r & r+1 & \ldots & n \end{array} \\ \begin{array}{c} 1 \\ \vdots \\ r \\ r+1 \\ \vdots \\ n \end{array} & \left[\begin{array}{c|c} \Sigma_1 & 0 \\ \hline 0 & \Sigma_2 \end{array}\right] \end{array} \tag{8}$$

15 Exercises

Exercise 80

$$\begin{array}{rcccccc} \|f\|^2 & = & e_1^2 u_1 \cdot u_1 & + & e_2^2 u_2 \cdot u_2 & + & e_3^2 u_3 \cdot u_3 \\ & & 18\bar{y}^2 & + & \mathtt{ss1} & + & \mathtt{ss2} \\ & & & & \multicolumn{3}{c}{\underbrace{104474 \quad + \quad 2284}_{106758}} \end{array}$$

Exercise 81

$$\begin{aligned} \operatorname{Cov}(U,V) &= (2,1) \begin{bmatrix} 1 & -1 \\ -1 & 4 \end{bmatrix} \begin{pmatrix} -2 \\ 1 \end{pmatrix} \\ &= (2,1) \begin{pmatrix} -3 \\ 6 \end{pmatrix} = 0 \end{aligned}$$

Exercise 82 *Typeset the matrix (8).*

16 Acknowledgements

While writing these tutorials I have had to expand my own knowledge of how to typeset Mathematics in LaTeX to cover topics that I had never really bothered with properly. I should like to thank David Carlisle, Frank Mittelbach and Chris Rowley for patiently answering my questions, even when they did not wholly approve of what I was writing. Of course, any remaining mistakes are my own, as are the personal opinions expressed.

Thanks also to many of my colleagues, both at Goldsmiths' College and at Queen Mary and Westfield College, for badgering me to explain how to do these things. And thanks to those readers who have sent kind messages of appreciation while the tutorials have been appearing.

References

[1] AMERICAN MATHEMATICAL SOCIETY: *AMS-LaTeX Version 1.2 User's Guide*, American Mathematical Society, Providence, Rhode Island, (1995).

[2] GOOSSENS, M., MITTELBACH, F. & SAMARIN, A.: *The LaTeX Companion*, Addison-Wesley, Reading, Mass., (1994).

[3] LAMPORT, L.: *LaTeX: A Document Preparation System*, first edition, Addison-Wesley, Reading, Mass., (1986).

[4] LAMPORT, L.: *LaTeX: A Document Preparation System*, second edition, Addison-Wesley, Reading, Mass., (1994).