

Typesetting Mathematics — User's Guide

B. W. Kernighan

and

L. L. Cherry

Bell Laboratories, Murray Hill, N. J.

ABSTRACT

This appendix is the user's guide for a system for typesetting mathematical expressions, using the phototypesetter on the UNIX operating system.

Mathematical expressions are described in a language designed to be easy to use by people who know neither mathematics nor typesetting. Enough of the language to set in-line expressions like $\lim_{x \rightarrow \pi/2} (\tan x)^{\sin 2x} = 1$ or display equations like

$$G(z) = e^{\ln G(z)} = \exp \left(\sum_{k \geq 1} \frac{S_k z^k}{k} \right) = \prod_{k \geq 1} e^{S_k z^k / k}$$

$$= \left(1 + S_1 z + \frac{S_1^2 z^2}{2!} + \dots \right) \left(1 + \frac{S_2 z^2}{2} + \frac{S_2^2 z^4}{2^2 \cdot 2!} + \dots \right) \dots$$

$$= \sum_{m \geq 0} \left(\sum_{\substack{k_1, k_2, \dots, k_m \geq 0 \\ k_1 + 2k_2 + \dots + mk_m = m}} \frac{S_1^{k_1}}{1^{k_1} k_1!} \frac{S_2^{k_2}}{2^{k_2} k_2!} \dots \frac{S_m^{k_m}}{m^{k_m} k_m!} \right) z^m$$

can be learned in an hour or so.

The language interfaces directly with TROFF, the phototypesetting language on UNIX, so mathematical expressions can be embedded in the running text of a manuscript, and the entire document produced in one process. This technical report is an example of its output.

The same language may be used with the UNIX formatter NROFF to set mathematical expressions on devices which have half-line forward and reverse motions, like model 37 Teletypes or GSI terminals.

Typesetting Mathematics — User's Guide

B. W. Kernighan & L. L. Cherry

1. Introduction

EQN is a UNIX program for typesetting mathematical material on the Graphics Systems phototypesetter. EQN works as a preprocessor for the typesetter formatter, TROFF, so the normal mode of operation is to prepare a document with both mathematics and ordinary text interspersed, and let EQN set the mathematics while TROFF does the body of the text.

The EQN language was designed to be easy to use by people who know neither mathematics nor typesetting. Thus EQN knows no mathematics. In particular, mathematical symbols like +, -, ×, parentheses, and so on have no intrinsic meaning. EQN is quite happy to set garbage (but it will look good).

Comments and criticisms on the equation-setter and this guide are solicited.

2. Usage

Equations are generally embedded in a larger text. To tell the equation-setter where an equation begins and ends, we mark it with lines beginning ".EQ" and ".EN". Thus, the lines

```
.EQ
x=y+z
.EN
```

produce

$$x = y + z$$

The ".EQ" and ".EN" are copied through untouched; they are not otherwise processed by the equation-setter. This means that you have to take care of things like centering, numbering, and so on yourself. To center an equation, for example, use

```
.ce
.EQ
a = b + c + d
.EN
```

There is also a shorthand notation useful for short in-line expressions, which is described in a later section.

To print a number of files containing equations on the phototypesetter, type

```
eqn filename(s) | troff
```

3. Spaces

Spaces and newlines in what you type are thrown away by the equation-setter. Thus

$$x = y + z$$

or

$$x = y + z$$

and so on all produce the same

$$x=y+z$$

Use spaces and newlines freely to make your input equations readable and easy to edit.

To force extra spaces into the *output*, use a tilde "~" for each space you want:

$$x\sim=y\sim+z$$

gives

$$x = y + z$$

You can also use a circumflex "^", which gives a space half the width of a tilde. It is mainly useful for fine-tuning. The circumflex is in all other respects identical to the tilde. Tabs may also be used to position pieces of an expression. The tabs stops must be set by TROFF commands.


4. Symbols, Special Names, Greek

The equation-setter knows some mathematical symbols, some mathematical names, and the Greek alphabet: (A complete list appears near the end of this Guide.)

$$x = 2 \pi \int \sin (\omega t) dt$$

produces

$$x=2\pi\int\sin(\omega t)dt$$

Here spaces are *necessary* in the input to tell EQN that "int", "pi", "sum", and "omega" are separate entities, that get special treatment. The "sin", digit 2, and parentheses are set in Roman type instead of italic; "pi" and "omega" are made Greek; and "int" becomes the integral sign. Knowledgeable users can also use TROFF four-character names for anything the equation setter doesn't know about, like \bs for the Bell System sign .

When in doubt, leave spaces around separate parts of the input.

5. Spaces, Again

The only way the equation-setter can deduce that some sequence of letters might be special is if that sequence is separated from the letters on either side of it. This can be done by surrounding a special word by ordinary spaces (or newlines), as we did in the previous section.

We can also make special words stand out by surrounding them with tildes or circumflexes:

$$x\sim=2\pi\int\sin(\omega t)dt$$

is much the same as the last example, except that the tildes not only separate the magic words like "sin", "omega", and so on, but also add extra spaces:

$$x = 2 \pi \int \sin (\omega t) dt$$

Special words can also be separated by braces { } and double quotes ", all of which have special meanings that we will see soon.

6. Subscripts and Superscripts

Subscripts and superscripts are available:

$$x \sup 2 + y \sub k$$

gives

$$x^2 + y_k$$

The equation-setter takes care of all size-changing and vertical motion. The words *sub* and *sup* may be in upper or lower case, as may most other words known to the equation-setter. Don't forget to leave a space (or its equivalent) to mark the end of a sub or superscript. A common error is to say something like

$$y = (x \text{ sup } 2)+1$$

which causes

$$y = (x^2)+1$$

instead of

$$y = (x^2)+1$$

Subscripted subscripts and superscripted superscripts also work:

$$x \text{ sub } i \text{ sub } 1$$

is

$$x_{i_1}$$

A subscript and superscript on the same thing are printed one above the other if the subscript comes *first*:

$$x \text{ sub } i \text{ sup } 2$$

is

$$x_i^2$$

Other than this special case, *sub* and *sup* group to the right, so

$$x \text{ sup } y \text{ sub } z$$

means

$$x \text{ sup } \{ y \text{ sub } z \}, \text{ not } \{ x \text{ sup } y \} \text{ sub } z$$

7. Using Braces for Grouping

Normally, the end of a subscript or superscript is marked simply by a blank (or tilde, etc.) What if the sub or superscript is something that has to be typed with blanks in it? Use { and } (braces) to mark the beginning and end of the sub or superscript:

$$x \text{ sup } \{ i \text{ omega } i \}$$

is

$$x^{i\omega i}$$

Rule: Braces { } can *always* be used to force the equation-setter to treat something as a unit, or just to make your intent perfectly clear. Thus:

$$x \text{ sub } \{ i \text{ sub } 1 \} \text{ sup } 2$$

is

$$x_{i_1}^2$$

with braces, but

$$x \text{ sub } i \text{ sub } 1 \text{ sup } 2$$

is

$$x_i$$

And braces can occur within braces if necessary:

$$x \sup (i \sup (\alpha + \beta))$$

is

$$x^{i^{\alpha + \beta}}$$

To print braces, enclose them in double quotes, like ". This is discussed in a later section.

8. Fractions

To draw a fraction, use the word *over*:

$$a+b \text{ over } 2c = 1$$

gives

$$\frac{a+b}{2c} = 1$$

The line is made the right length and positioned automatically. Braces can be used to make clear what goes over what:

$$\{\alpha + \beta\} \text{ over } \{\sin(x)\}$$

is

$$\frac{\alpha + \beta}{\sin(x)}$$

Over is considered by the equation-setter to be of lower precedence than *sub* and *sup*, so

$$-b \sup 2 \text{ over } \{\pi x\}$$

needs no braces to be unambiguously

$$\frac{-b^2}{\pi x}$$

The precedence rules, which decide which operation is done first, are summarized near the end of the user's guide. When in doubt, however, *use braces* to make clear what goes with what.

9. Square Roots

To draw a square root, use *sqrt*:

$$\text{sqrt } a + \text{sqrt}(a x \sup 2 + b x + c)$$

is

$$\sqrt{a} + \sqrt{ax^2 + bx + c}$$

Warning — square roots of tall quantities look lousy, because a root-sign big enough to cover the quantity is too dark and heavy:

$$\text{sqrt}(a \sup 2 \text{ over } b \text{ sub } 2)$$

is

$$\sqrt{\frac{a^2}{b_2}}$$

Good style replaces big square roots by something to the power $\frac{1}{2}$:

$$(a^2/b_2)^{1/2}$$

10. Summation, Integral, Etc.

Summations, integrals, and similar constructions are easy:

sum from $i=0$ to $\{i=\text{inf}\}$ x sup i

produces

$$\sum_{i=0}^{\text{inf}} x^i$$

Notice that we used braces to indicate where the upper limit begins and ends. No braces were necessary for the lower limit, because it contained no blanks. The braces will never hurt, though — *always* use braces around the *from* and *to* limits, if they contain any blanks.

The *from* and *to* parts are both optional, but if both are used, they have to be in that order.

Other useful characters can replace the “sum”:

int prod union inter

become, respectively,

$$\int \prod \cup \cap$$

Since the thing before the *from* can be anything, the from-to construction can often be used in unexpected ways:

lim from $\{n \rightarrow \text{inf}\}$ x sub $n=0$

is

$$\lim_{n \rightarrow \infty} x_n = 0$$

11. Big Brackets, Etc.

To get big brackets [], braces { }, parentheses (), and bars || around things, use the *left* and *right* commands:

left { a over b + 1 right } = left (c over d right) + left [e right]

is

$$\left\{ \frac{a}{b} + 1 \right\} = \left(\frac{c}{d} \right) + [e]$$

The resulting brackets are made big enough to cover whatever they enclose.

Several warnings about brackets are in order. First, notice that braces are typically bigger than brackets and parentheses, because they are made up of three, five, seven, etc. pieces, while brackets can be made up of two, three, etc. Second, left and right parentheses typically look poor, because the character set is poorly designed. Finally, the “right” part may be omitted: a “left something” need not have a corresponding “right something”. If the *right* part is omitted, put braces around the thing you want the left bracket to encompass. Otherwise, the resulting brackets may be too large.

12. Piles

There is a general facility for making vertical piles of things; it comes in several flavors. For example:

A “=” left [

pile { a above b above c } “=” pile { x above y above z }

right]

will make

$$A = \begin{bmatrix} a & x \\ b & y \\ c & z \end{bmatrix}$$

The elements of the pile (there can be as many as you want) are centered one above another, at the right height for most purposes. The keyword *above* is used to separate the pieces; braces are used around the entire list.

Three other forms of pile exist: *lpile* makes a pile with the elements left-justified; *rpile* makes a right-justified pile; and *cpile* makes a centered pile, just like *pile*. The vertical spacing between the pieces is somewhat larger for l-, r- and cpiles than it is for ordinary piles.

`sign(x) = left { lpile { 1 above 0 above -1 }`
`-- lpile { if x > 0 above if x = 0 above if x < 0 }`

makes

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Notice the left brace without a matching right one.

13. Size and Font Changes

By default, equations are set in 10 point type, with standard mathematical font conventions. Although the equation-setter makes a valiant attempt to use esthetically pleasing sizes and fonts, it is not perfect. To change sizes and fonts, use *size n* and *roman*, *italic*, and *bold*. Legal sizes which may follow *size* are 6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24, 28, 36.

`size 14 bold x = y + size 14 { alpha + beta }`

gives

$$x = y + \alpha + \beta$$

As always, use braces to delimit what you want affected. For example, you can change an entire equation by

`size 12 { ... }`

14. Diacritical Marks

To get funny marks on top of letters:

`x dot + X dot + y hat + y doidot + x-y bar + {alpha + beta} bar + x tilde`

gives

$$\dot{x} + \dot{X} + \hat{y} + \ddot{y} + \overline{x-y} + \overline{\alpha+\beta} + \tilde{x}$$

As well as possible, the mark is placed at the right height. The *bar* is made the right length for the entire construct; other marks are centered. (At present, this works only on italic letters; other fonts are botched.) By the way, there is no "prime" — use "'".

15. Quoted Text

Any input entirely within quotes ("...") is not subject to any of the font changes and spacing adjustments normally done by the equation setter. This provides a way to do your own spacing and adjusting if needed:

italic "sin(x)" + sin (x)

is

sin(x)+sin(x)

Notice the differences. Quotes are also used to get braces and other EQN keywords printed:

"{ alpha }"

is

{ alpha }

To get a quote into a quoted string, use "\".

16. Shorthand for In-line Equations

In a mathematical document, it is necessary to follow mathematical conventions not just in display equations, but also in the body of the text, for example by making variable names italic. Although this can be done by surrounding the appropriate parts with ".EQ" and ".EN", the continual repetition of ".EQ" and ".EN" is a nuisance.

EQN provides a shorthand for short in-line sequences. You can define two characters to mark the left and right ends of an in-line equation, and then type expressions in the middle of text lines. To set the left and right characters to dollar signs, for example, use

```
.EQ
delim $$
.EN
```

Having done this, you can then say things like

Let \$alpha sub i\$ be the primary variable, and let \$beta\$ be zero. Then we can show that \$x sub i\$ is \$>=0\$.

This works as you might expect — spaces, newlines, and so on are significant in the text, but not in the equation part itself. Multiple equations can occur in a single input line.

To turn off the delimiters,

```
.EQ
delim off
.EN
```

Warning: don't use braces, tildes, circumflexes, or double quotes as delimiters — chaos will result.

17. Definitions

EQN provides a facility so you can give frequently used string of characters a name, and thereafter just type the name instead of the whole string. For example, if the sequence

$x_{i-1} + y_{i-1}$

were to appear repeatedly throughout a memo, you can save re-typing it each time by defining it like this:

```
.EQ
define xy "x sub i sub 1 + y sub i sub 1"
.EN
```

and then using it like this:

```
.EQ
f(x) = xy ...
```


.EN

and so on. The "define" statement makes "xy" a shorthand for the longer expression. Be careful to leave spaces or their equivalent around the name, so EQN will be able to identify it as special.

There are several warnings: First, although definitions can use previous definitions, as in

```
.EQ
define xi "x sub i"
define xil "xi sub l"
.EN
```

don't define something in terms of itself. Second, although EQN keywords can be redefined, this should be done with some care. Although you can make "/" mean "over" by saying

```
define / "over"
```

to redefine "over" as "/", you have to write

```
define "over" "/"
```

to avoid a syntax error.

18. A Large Example

Here is the complete source for the three display equations on the cover sheet of this memo:

```
.nf
.in li
.sp 5p
.EQ
G(z) ~ e sup { ln ~ G(z) }
~ exp left (
sum from k >= 1 { S sub k z sup k } over k right )
~ prod from k >= 1 e sup { S sub k z sup k / k }
.EN
.sp
.EQ
----- = left ( 1 + S sub 1 z +
{ S sub 1 sup 2 z sup 2 } over 2! + ... right )
left ( 1 + { S sub 2 z sup 2 } over 2
+ { S sub 2 sup 2 z sup 4 } over { 2 sup 2 cdot 2! }
+ ... right ) ...
.EN
.sp
.EQ
----- = sum from m >= 0 left (
sum from
pile { k sub 1 . k sub 2 ..., k sub m >= 0
above
k sub 1 + 2k sub 2 + ... + mk sub m = m }
{ S sub 1 sup { k sub 1 } } over { 1 sup k sub 1 k sub 1 ! } -
{ S sub 2 sup { k sub 2 } } over { 2 sup k sub 2 k sub 2 ! } -
...
{ S sub m sup { k sub m } } over { m sup k sub m k sub m ! }
right ) z sup m
.EN
```

.fi
.in 0
.sp 5p

19. Keywords, Precedences, Etc.

Much of this section can be skipped on first reading.

The keywords in the next two lists are recognized in upper or lower case. These keywords have no precedence associated with them:

lpile cpile rpile pile above define delim

These keywords are listed in order of increasing precedence:

from to
over sqrt sup sub
size roman italic bold
left right
dot dotdot hat tilde bar

These group to the left:

over sqrt left right

All others group to the right.

Digits, parentheses, brackets, punctuation marks, and these mathematical words are converted to Roman font when encountered:

sin cos tan arc lim max min log ln exp

These character sequences are recognized:

>= <= +- == -> <- != ... ,....

sum int prod union inter inf partial times cdot

and translated to, respectively,

≥ ≤ ± ≡ → ← ≠ ... ,....

$\Sigma \int \Pi \cup \cap \infty \partial \times$

Finally, those Greek letters which are not identical to some Roman letter are known; simply spell them out in whatever case you want:

alpha gamma GAMMA

is

$\alpha \gamma \Gamma$

Notice that GAMMA is all capitals.

20. Implementation Peculiarities and Other Blemishes

When you leave an equation, the equation-setter restores you to the size and font in which you entered.

TROFF number-registers and string macros numbered 10 through 99 are used. Avoid them.

Equations may be set in-line, rather than in display; in fact this is the default mode, since ".EQ" and ".EN" are not defined by the equation setter, but are simply copied from input to output. Enough room is left before and after the line containing an equation so that it does not in-

interfere with other lines, so an in-line expression like $\text{erf}(x) = \int_0^x e^{-t^2} dt$ does not interfere with the lines surrounding it. This was produced by

roman erf (x)= int size 9{~ from 0 to x } e sup -t sup 2 dt

In-line equations can only be so big because of an internal buffer in TROFF. If you get a message "word overflow", you have exceeded this limit. If you print the equation in no-fill mode this message will usually go away. The message "line overflow" indicates you have exceeded an even bigger buffer in no-fill mode. The only cure for this is to re-write the equation.

On a related topic, EQN does not break equations by itself — you must split long equations up across multiple lines by yourself, marking each by a separate ".EQEN" sequence. If operators must be lined up vertically, use tildes to get to the right horizontal position — examine the big examples in an earlier section.

EQN occasionally fails to give the right spacing on some construction. If you feel you have to tune things, and are willing to use some TROFF, you can put in your own local motions. The best way is to enclose them in double quotes.

21. Use on Model 37 Teletypes and GSI Terminals

A compatible version of EQN can be used on devices like teletypes and GSI terminals which have half-line forward and reverse capabilities. To print equations on a model 37 teletype, for example, use

neqn files | nroff

The language for equations recognized by NEQN is identical to that of EQN, although of course the output is more restricted.

To use a GSI terminal as the output device,

neqn files | nroff | gsi

22. Acknowledgments

We are deeply indebted to J. F. Ossanna, the author of TROFF, for his willingness to extend TROFF to make our task easier, and for his continuous assistance during the development of the EQN program. We are also grateful to A. V. Aho for advice on language design, and to S. C. Johnson for assistance with the YACC compiler-compiler.

EQN changes

Minor:

1. Underbars can be made with the keyword "under"; it is quite analogous to "bar".
2. The keyword "approx" gives the double wiggly line previously produced by " \approx ".
3. Error messages now come out on the error file instead of through troff, so you can pre-check a big document by flushing the output down /dev/null. Thus:

```
eqn file ... >/dev/null
```

checks for errors.

4. Superscripts and subscripts are now better positioned when there is a tiny one on a big object.

Major:

1. Equations can now be lined up with previous ones. The keyword "mark" remembers the current position; in a later equation you can use "lineup" to force that position to line up with the previously marked place. Thus:

```
.EQ
f(x) mark = y
.EN
.EQ
lineup = z
.EN
```

lines up the equals signs.

2. By popular demand, matrices have arrived (at great expense to the management, I might add). Matrices are introduced by the keyword "matrix". Elements are specified by column in much the same way as piles. EACH COLUMN MUST HAVE THE SAME NUMBER OF ELEMENTS, or I will not be responsible for the results. Each column may be individually left-justified, centered, or right-justified. Matrices may be nested, although surely no one would want to. Thus:

```
.EQ
matrix { lcol { a above b } ccol { c above d } rcol { e above f } }
.EN
```

That is, columns are "lcol", "ccol", or "rcol"; a column is just "ccol { ... above ... above ... }", and a matrix is just "matrix { column column ... }". Be sure to get the right number of braces. But notice that there is no need for braces around the things that lie between "above" keywords.

There is no decent way to get extra space between the columns.