Basic Mathematics

BY T. P. JONES, O.L.S.

THE ROUTINES for generating a sine and cosine are listed in Figure 1. Angular data for these sub-routines, which have been written to produce trig. functions to double precision, should be in degrees and decimal degrees, and not in radians.

Because all Basic languages are not alike, some of the statements may appear to be overly long and complicated. For example, If A# is less than zero, then A# = A# ⋆ -1 is used merely to generate the absolute value of A#. Some computer functions reach an absolute value by first squaring the argument, and then extracting the square root of the square. If the square root so extracted is only to single precision, then the eventual accuracy will not be attained.

If your machine does give an absolute value to double precision, then this statement could be changed to A# = ABS(A#), or whatever.

These routines will run slower than library functions which perform the same operations. This is because every statement in the routine has to be translated from Basic to machine language before being executed, and the translation is carried out every time the statement is read. There are several statements which are repeated over and over again until the increment reaches zero, and they are translated every time. A library function, on the other hand, is already in machine language and can utilize the full speed of the computer.

NOTE

While the routine listed here has undergone extensive testing to be sure that it operates properly, neither myself nor the Editorial Board make an express or implied warranty of any kind with regard to it. In no event will we, or The Association of Ontario Land Surveyors, be liable for incidental or consequential damages in connection with or arising out of the furnishing, performance or use of any of these programs.

As mentioned before, the argument must be ZO#, and the trig. function computed will be ZJ#.

Murphy, and his law, appeared to be working overtime on the last issue; for this reason, the square root routine is repeated here in Figure 1. Also included is a simple program to access and test the sine and cosine sub-routines. ●

```
10     INPUT N#:  ZO# = N#:  GOSUB 40215
20     PRINT N#, ZJ#,:  ZO# = N#:  GOSUB 40245
30     PRINT ZJ#:  GOTO 10
40060 REM
40065 REM    >>> SQUARE-ROOT ROUTINE <<<
40070 REM
40075 ZL# = ZO#:  ZJ# = ZL#:  ZK# = 0
40080 IF ZJ# <> ZK# THEN ZK# = ZJ#:  ZJ# = (ZJ# + ZL# / ZK#) / 2:  GOTO 40080
40085 RETURN
40200 REM
40205 REM    >>> SINE ROUTINE <<<
40210 REM
40215 ZO# = ZO# / 57.29577951308232
40220 ZT% = 0:  IF ZO# < 0 THEN ZO# = ZO# * -1:  ZT% = 4
40225 GOTO 40255
40230 REM
40235 REM    >>> COSINE ROUTINE <<<
40240 REM
40245 ZO# = ZO# / 57.29577951308232
40250 ZT% = 2:  IF ZO# < 0 THEN ZO# = ZO# * -1
40255 ZC# = 0.7853981633974483:  ZO# = ZO# / ZC#:  ZR% = INT(ZO#)
40260 ZO# = (ZO# - ZR%) * ZC#:  ZR% = ZR% + ZT%
40265 ZP% = INT(ZR% / 8) * 8:  ZR% = ZR% - ZP%:  ZT% = ZR%
40270 ON (ZT% + 1) GOTO 40280, 40275, 40280, 40275, 40280, 40275, 40280, 40275
40275 ZO# = ZC# - ZO#
40280 IF ZT% > 3 THEN ZU% = -1 ELSE ZU% = 1
40285 ON (ZT% + 1) GOTO 40295, 40290, 40290, 40295, 40295, 40290, 40290, 40295
40290 ZQ% = -1:  ZL# = 1:  ZJ# = 1:  GOTO 40300
40295 ZQ% = 1:  ZL# = ZO#:  ZJ# = ZO#
40300 ZK# = ZO# * ZO#:  ZP% = 0:  ZS% = -1
40305 ZP% = ZP% + 2:  ZQ% = ZQ% + 2:  ZL# = ZL# * ZK# * ZS% / (ZP% * ZQ%)
40310 IF ZL# <> 0 THEN ZJ# = ZJ# + ZL#:  GOTO 40305
40315 ZJ# = ZJ# * ZU%:  RETURN
```