CONVERSION OF ANGULAR QUANTITIES

N. Vučetić, S. Petrović, N. Frančula and M. Lapaine

Geodetic Faculty, University of Zagreb, Yugoslavia

ABSTRACT

The paper offers a simple and reliable algorithm for conversions of angular quantities between various forms of their representation.

INTRODUCTION

When performing geodetic calculations by computers, it is no wonder that the same angular quantity often appears represented in several different forms. For instance, such data may be given and entered in degrees, minutes and seconds, but to calculate trigonometric functions (in FORTRAN or BASIC) one has to convert them to radians. At first, such conversions might seem simple and easy. However, as pointed out for example in [1] the algorithms which look completely reasonable may fail in some cases, and result in unacceptable errors. Further, the subprogram for the conversion of radians to degrees, minutes and seconds, published in [2], gives for example 36° - l' 59" in place of 35° 59' 59", which is obviously not a very suitable form for printing the results of computations. We offer the following alternative simple, but reliable solution (implemented in FORTRAN).

CONVERSIONS

Let us consider the following forms of angular quantities:

1. The representation of degrees, minutes and seconds (e.g. 39° 23' 43.67") as a single number (39.234367), symbolised by DD.MMSS in the following text.
2. Decimal degrees (39° 23' 43.67" represented as 39.3954639), denoted by D.
3. Degrees, minutes and seconds as 3 separate numbers (39, 23, 43.67), denoted by D, M, S;
4. Radians (0.687580555), denoted by *R.*

The scheme of the possible necessary conversions is:

$$\mathrm{DD.MMSS} \underset{6}{\overset{1}{\rightleftharpoons}} \mathrm{D, M, S} \underset{5}{\overset{2}{\rightleftharpoons}} \mathrm{D} \underset{4}{\overset{3}{\rightleftharpoons}} \mathrm{R.}$$

We restrict our discussion to the case of positive angles. Namely, the negative sign should be taken into account separately (for example, assigning it to degrees when using the D, M, S form, would result in the identification of 0° 15' 10" with — 0° 15' 10", which is obviously not acceptable). The conversions 2, 3, 4 and 6 are now straightforward translations of well known mathematical relationships, and cause no difficulties. The problems connected with the remaining two are the consequence of two facts: the binary representation of numbers in computers, and

the finite number of (binary) digits used for that representation. The solution lies in an unusual application of the rounding-up technique, which reverts the order of operations. Namely, the quantity needed in order to ensure the correct rounding of the seconds (or their decimal fractions) is added first, while stripping off degrees and minutes follows only after that.

On the basis of those principles, the conversion 1 (DD.MMSS → D, M, S) can be performed by the following FORTRAN subroutine:

```
C CONVERT DD. MMSS TO DEG, MIN, SEC
      SUBROUTINE DEMISE(ROU, X, ID, IM, SE)
      DOUBLE PRECISION ROU, X, HLP, DM, SE
      INTEGER*4 ID, IM,IHLP
      HLP = X + IDO / (20000D0*ROU)
      ID = HLP
      DM= (HLP —ID) *100D0
      IM = DM
      IHLP = (DM—IM) *100D0*ROU
      SE = IHLP/ROU
      RETURN
      END
```

Here we use the FORTRAN built-in conversions between double precision reals and long integers. The starting value of the form DD.MMSS should be transferred to the variable X, while the resulting 3 separate values for degrees, minutes and seconds are ID, IM and SE. The factor 20000 reflects the fact that 0.5" is in the form DD.MMSS represented as 0.00005 = 1/20000.

The rounding factor ROU should be chosen in accordance with the precision offered by the computer's internal representation of numbers. Suppose that we use standard 8-byte double precision reals with 4-byte long integers, the guaranteed precision amounts to 1/10000000" in the whole range from 0° to 360°. It means that the largest permissible value for ROU is 10000000. Since, in practice, one usually does not calculate with such a precision, a smaller value of ROU can be applied as well. For instance, to achieve 1/10", we can choose any of the values 10, 100, 1000,..., 10000000. If we have no doubt which is the maximal permissible value of ROU, it is natural to chose that value. A less experienced user, wanting to implement our subprogram on a machine having a different length of internal number representation, will be more safe to use ROU = 10 when all of his data are accurate and given up to 1/10". Similarly, ROU = 100 will guarantee 1/100", and so on, assuming, of course, that it is possible to achieve that precision with the given number of internally used (binary) digits.

When we wish to use the standard FORTRAN 4-byte single precision reals and 2-byte integers, we simply have to change the variables type declarations in the proposed subroutine. This guarantees no more than a l" resolution in the whole range from 0° to 360°, and ROU should be set equal to 1.

We have tested the proposed algorithm practically for the single precision case

from 0° 0' 0" to 359° 59' 59" with a 1" step, both in FORTRAN 77 under MS DOS and FORTRAN 80 under CP/M. We could not make such detailed tests for the various double precision cases (1/10", 1/100",...), as it would take too long.However, we have checked what we consider to be the most critical cases. The proposed algorithm did not fail on any example.

It should be noticed that the same subroutine solves the problem of the conversion 5 (D → D, M, S). One only needs to replace every 100 by 60, and 20000 by 7200.

CONCLUSION

The proposed algorithm for the conversion of angular quantities proved to be simple and reliable. The FORTRAN version is included in the present paper, and the translation to other programming languages should not cause any special difficulties. It may only require the explicit use of some form of INT function. A ready for use BASIC version of the algorithm can be found in our former, more detailed paper [3], which dealt also with other fundamental algorithms relating to geodetic calculations.

References

1. King, C. W. B., 1988. Computational formulae for the Lambert conformal projection (Part 2). *Survey Review,* XXIX (230); 387-399.
2. Newton, G. D., 1985. Computer Programs for Common Map Projections. *U.S. Geological Survey Bulletin,* 1642; 1-33.
3. Vučetić, N., Petrović, S., Lapaine, M. & Frančula, N., 1988. Prijedlozi za standardizaciju nekih osnovnih algoritama u geodetskim računanjima (Proposals for the standardization of some fundamental algorithms in geodetic calculations, croato-serbian). *Geodetski list,* LXV(4-6); 159-165.