Migration with Short Computer Word Length

by Bjorn Engquist

The error in migration from round-off during the computation is in general much smaller than other errors. The usual computer word is unnecessarily long. For the use of minicomputers and in general to reduce storage and I/O, one could try shorter words when it is practically possible.

Here are the results from some experiments where a short mantissa (fraction) in floating point arithmetic was used. The calculations in the inner loop were performed in full accuracy (24 bit mantissa on IBM 370/168). The values of the wave field were then truncated or rounded before they were stored. This amounts to introducing rounding errors for each new t and z level.

The tests show that a considerable reduction of the length of the mantissa can be done without distorting the migration. In some cases it can be as short as 5-7 bits.

We used the wave equation program SM8 (SEP-8, p. 118) which is based on an implicit difference approximation of $P_{tz} = \frac{v}{2} P_{xx}$. The program was modified to include truncation and rounding, and we had a symmetric pulse with maximum 1 located at the left boundary as initial value. All computations were performed on IBM 370/168 with hexadecimal arithmetic. The mantissa of a single precision floating number P was truncated or rounded by the fortran functions TRUNC or ROUND, which are listed after the figures. With truncation

we mean chopping of the n least significant bits of the mantissa. If P is rounded we do the same thing, but if the chopped off part is $\geq 2^{n-1}$ we add 2^n to the mantissa if we regard it as an integer. We also simulated binary truncation and rounding by the functions BINTR and BINRO. The leading digit d_1 in the hexadecimal mantissa has the range $1 \leq d_1 \leq F$, which means a waste of 1 to 4 bits when compared to a standard binary representation. (The exponent can of course be shorter in hexadecimal representation.) In the figure captions TR stands for truncation and RO for rounding. Then follows the number of bits in the mantissa and within parentheses the base for the exponent. The number of steps in the z direction are denoted by NZ .

We see in Figure 1 that 3 hexadecimal digits (12 bits) and rounding are enough to give a plot which cannot be distinguished from the one produced by full word length calculation (Figure 4d). Truncation to 3 digits is also almost satisfactory but gives larger errors than rounding and for this mainly positive wave form it results in a lower magnitude.

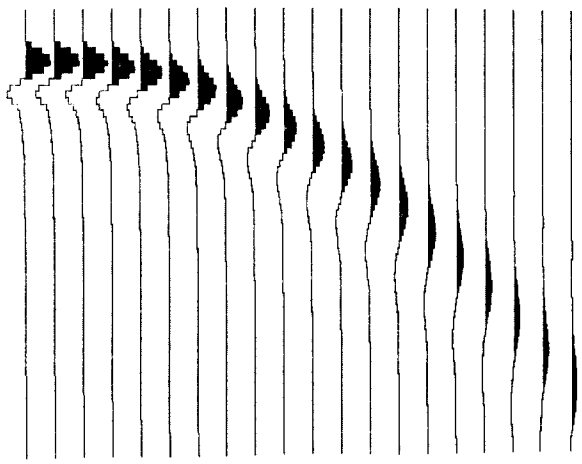
The binary representation requires fewer bits in the mantissa. This can be seen in Figure 2. The critical number of bits is here around 7. In Figure 3 we study different numbers of z iterations. We are on the border of indistinguishable results, but it is possible to see that truncation to 9 bits does a better job than rounding to 7 bits for $NZ = 25$. The situation is reversed when we iterate 100 steps in z . It is very difficult to strictly analyze the

statistical properties of truncation and rounding errors in the migration process.

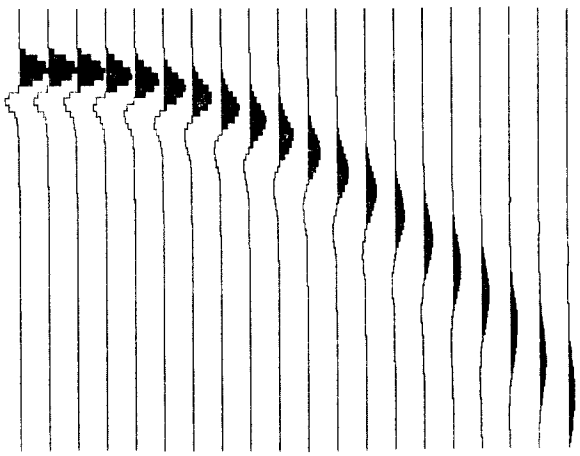
Let us for simplicity assume that the result of truncating P is uniformly distributed in $(P - \varepsilon, P)$ and the corresponding result for rounding is in $(P - \varepsilon/2, P + \varepsilon/2)$, and that the errors from different iterations are just added. This would cause the error to grow like $O(NZ)$ when truncating and like $O(NZ^{1/2})$ when rounding. It is consistent with the result presented in Figure 3. The stability of the difference equation guarantees that the error does not grow exponentially with NZ.

In the Figures 4a and 4b we see that truncating to 4 hexidecimal digits or to 11 binary ones is enough for $NZ = 50$. The plot 4c shows that rounding also in the inner loops of the calculations does not distort the final result very much.

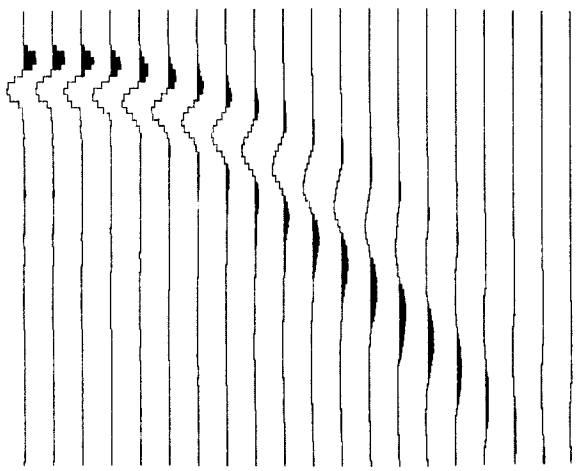
In these tests the full 7 bit hexadecimal exponent was used. Such a large exponent range is of course not needed.



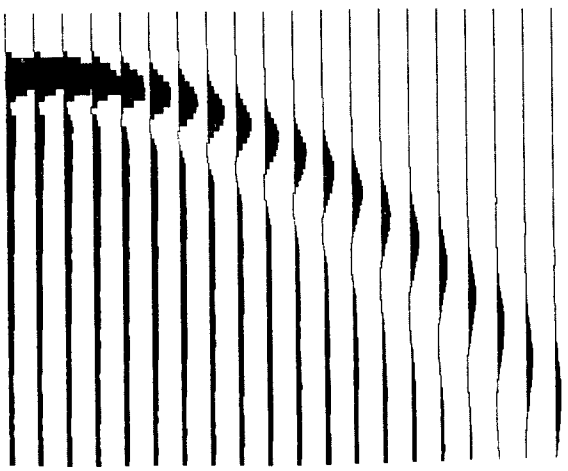
a: TR 12 (16), NZ = 50



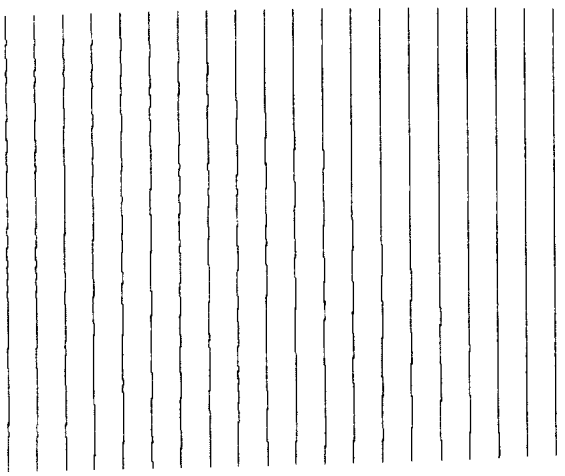
b: RO 12 (16), NZ = 50



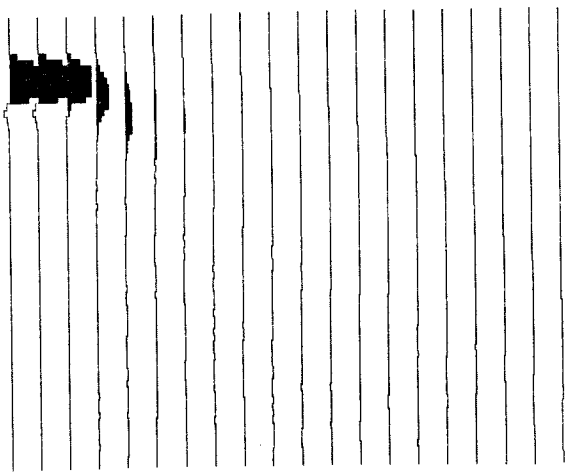
c: TR 8 (16), NZ = 50



d: RO 8 (16), NZ = 50

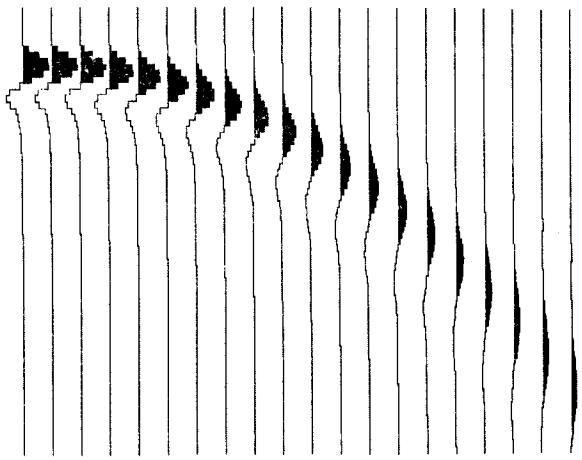


e: TR 4 (16), NZ = 50

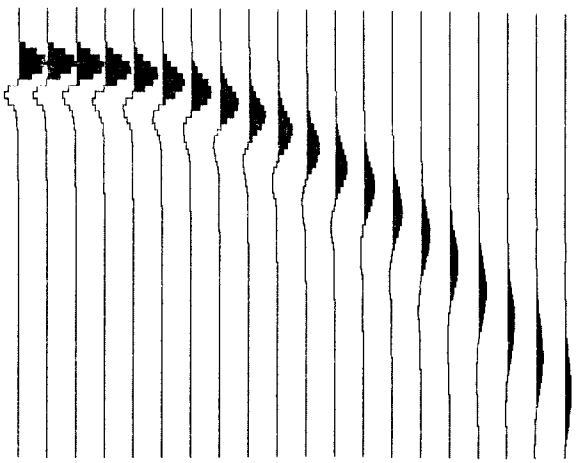


f: RO 4 (16), NZ = 50

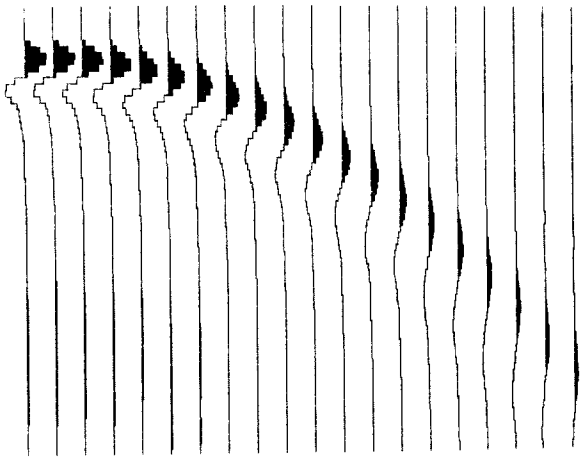
Figure 1 a - f .



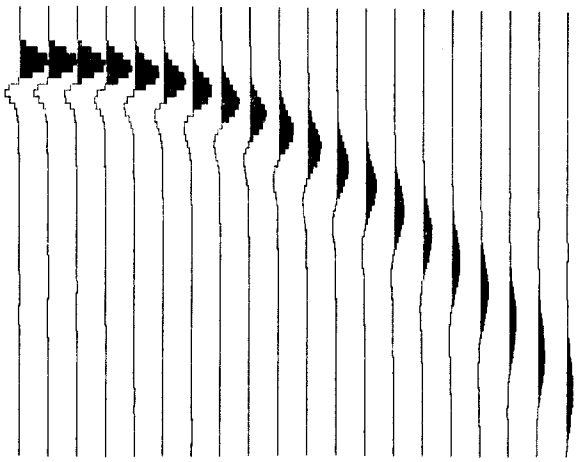
a: TR 9 (2), NZ = 50



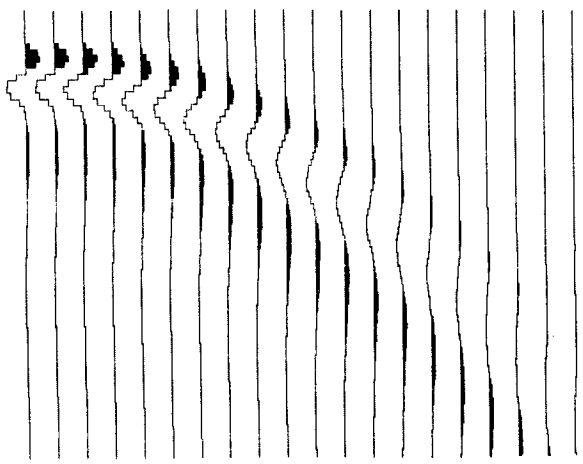
b: RO 9 (2), NZ = 50



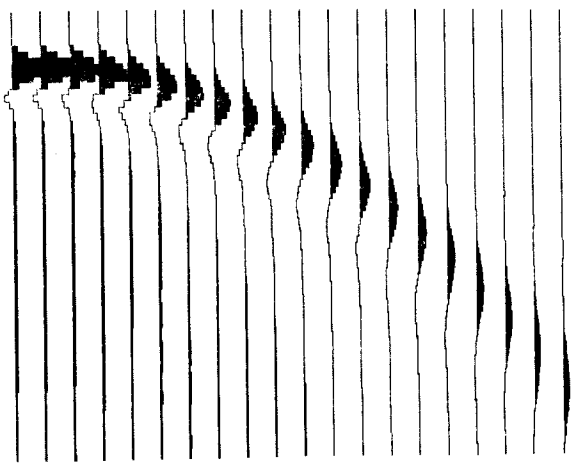
c: TR 7 (2), NZ = 50



d: RO 7 (2), NZ = 50

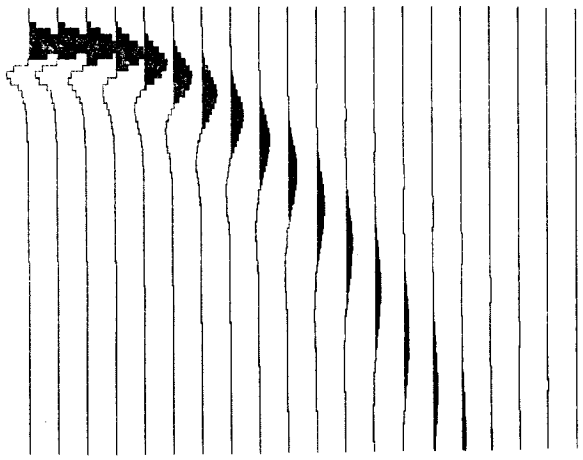


e: TR 5 (2), NZ = 50

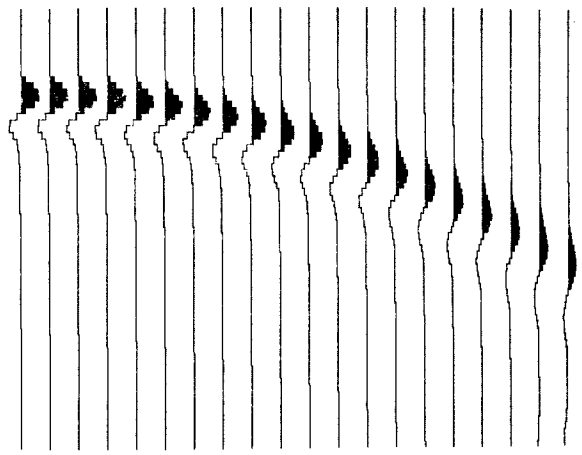


f: RO 5 (2), NZ = 50

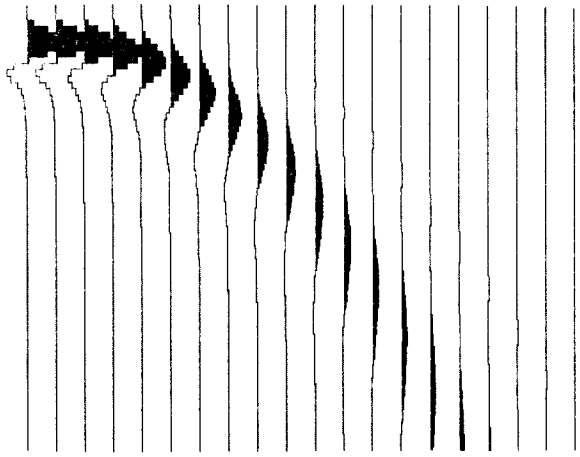
Figure 2 a - f .



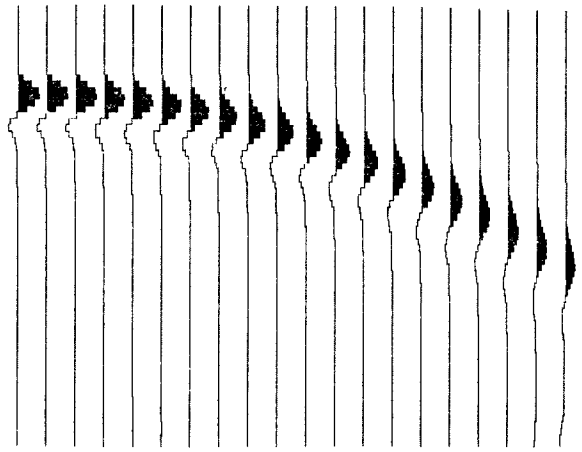
a: TR 9 (2), NZ = 25



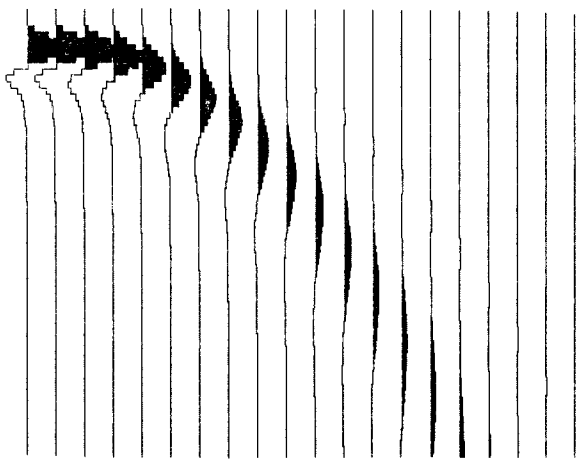
b: TR 9 (2), NZ = 100



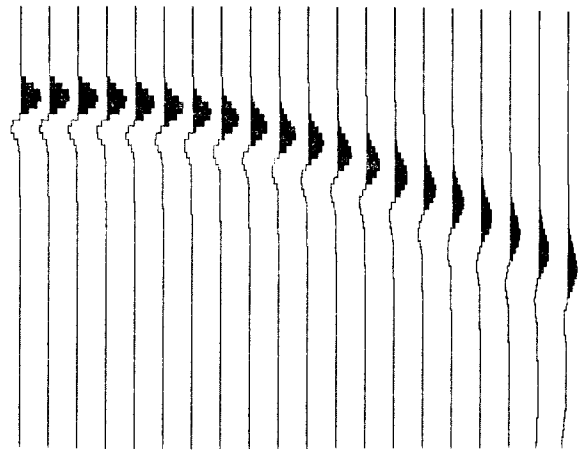
c: RO 7 (2), NZ = 25



d: RO 7 (2), NZ = 100

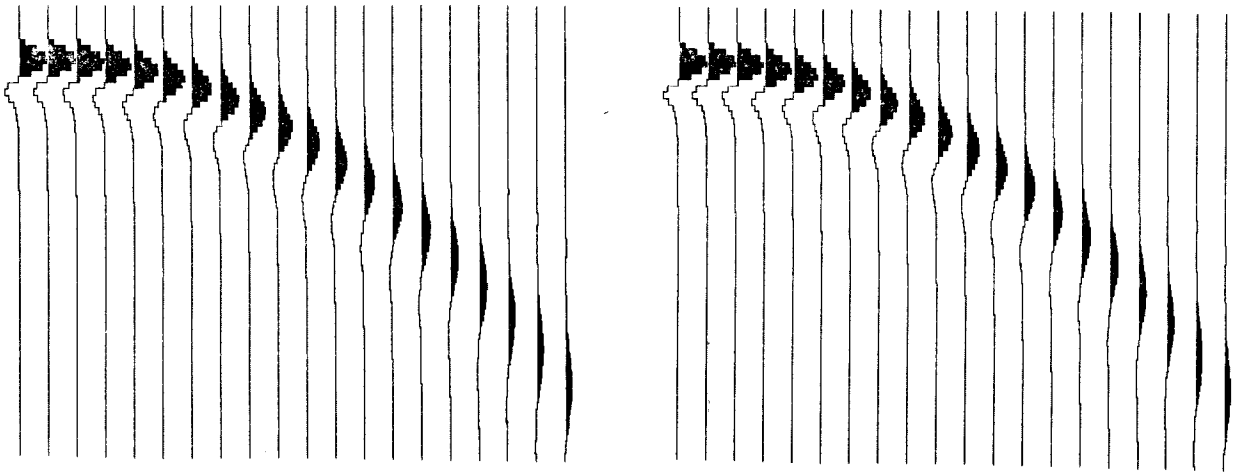


e: 24 bits, NZ = 25



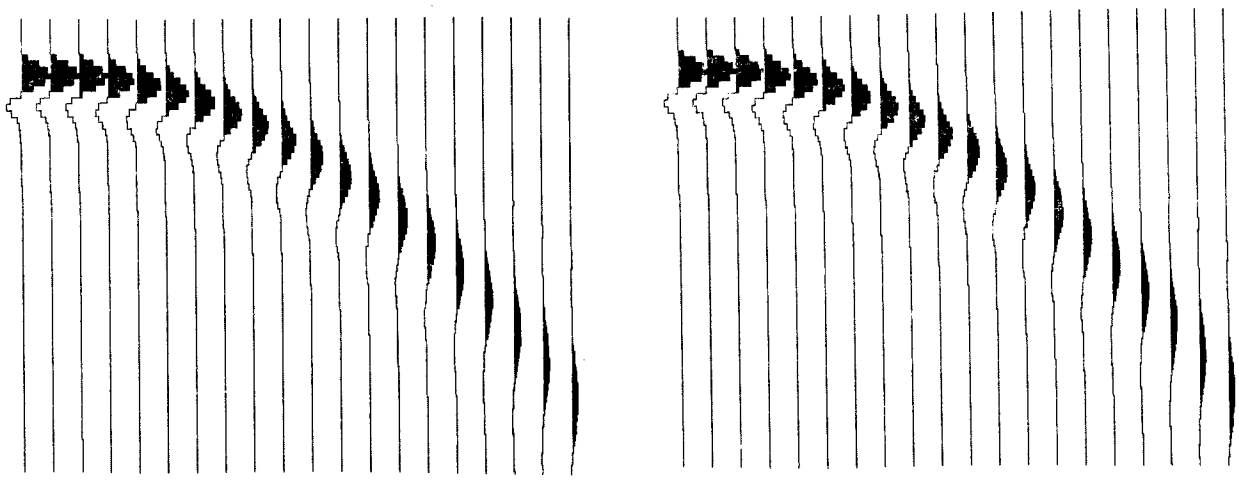
f: 24 bits, NZ = 100

Figure 3 a - f .



a: TR I6 (I6), NZ = 50

b: TR 11 (2), NZ = 50



c: RO[1)] 7 (2), NZ = 50

d: 24 bits, NZ = 50

Figure 4 a - d .

1) In this test we rounded to a 7 bit mantissa also in the inner loop (x loop) and in the tridiagonal solver.

```
      FUNCTION TRUNC(X)
C***      X IS TRUNCATED NR BITS FROM THE RIGHT. M = 2**NR
      COMMON /MM/ NR,M
      EQUIVALENCE (Y,I)
      Y = ABS(X)
      I = M*(I/M)
      IF(X.LT.0.) Y=-Y
      TRUNC = Y
      RETURN
      END

      FUNCTION ROUND(X)
C***      X WITH A 24 BIT MANTISSA IS ROUNDED NR BITS FROM
C***      THE RIGHT. M = 2**NR
      COMMON /MM/ NR,M
      EQUIVALENCE (Y,I)
      DATA M23,M24 /8388608,16777216/
      Y = ABS(X)
      IF(I-M24*(I/M24).LT.M23) GOTO 10
      Y = Y/2.
      M = M/2
      I = M*((I+M/2)/M)
      M = 2*M
      IF(X.LT.0.) Y=-Y
      ROUND = 2.*Y
      RETURN
   10 I = M*((I+M/2)/M)
      IF(X.LT.0.) Y=-Y
      ROUND = Y
      RETURN
      END

      FUNCTION BINTR(X)
C***      X WITH A 24 BIT MANTISSA IS TRUNCATED NR BITS FROM
C***      THE RIGHT, SIMULATING BINARY REPRESENTATION WITH
C***      HIDDEN BIT NORMALIZATION. X IS REPRESENTED IN
C***      HEXADECIMAL FORM. M = 2**NR
      COMMON /MM/ NR,M
      EQUIVALENCE (Y,I)
      DATA M22,M23,M24 /4194304,8388608,16777216/
      Y = ABS(X)
      J = I-M24*(I/M24)
      ME = M/16
      IF(J.LT.M22) GOTO 99
      ME = ME*2
      IF(J.LT.M23) GOTO 99
      ME = ME*2
      IF(J.LT.M24) GOTO 99
      ME = ME*2
   99 I = ME*(I/ME)
      IF(X.LT.0.) Y=-Y
      BINTR = Y
      RETURN
      END
```

```
      FUNCTION BINRO(X)
C***      X WITH A 24 BIT MANTISSA IS ROUNDED NR BITS FROM
C***      THE RIGHT, SIMULATING BINARY REPRESENTATION WITH
C***      HIDDEN BIT NORMALIZATION. X IS REPRESENTED IN
C***      HEXADECIMAL FORM.  M = 2**NR
      COMMON /NR/ NR,M
      EQUIVALENCE (Y,I)
      DATA M22,M23,M24 /4194304,8388608,16777216/
      Y = ABS(X)
      J = I-M24*(I/M24)
      ME = M/16
      IF(J.LT.M22) GOTO 99
      ME = ME*2
      IF(J.LT.M23) GOTO 99
      ME = ME*2
      IF(J.LT.M24) GOTO 99
      ME = ME*2
 99   IF(I-M24*(I/M24).LT.M23) GOTO 10
      Y = Y/2.
      ME = ME/2
      I = ME*((I+ME/2)/ME)
      ME = 2*ME
      IF(X.LT.0.) Y=-Y
      BINRO = 2.*Y
      RETURN
 10   I = ME*((I+ME/2)/ME)
      IF(X.LT.0.) Y=-Y
      BINRO = Y
      RETURN
      END
```