Subject: Re: Inaccuracies
Posted by wclodius on Tue, 14 Nov 1995 08:00:00 GMT
View Forum Message <> Reply to Message

Some points:

1. It is impossible in finite binary arithmetic to represent any fraction that is not an integer multiple of a power of two. The fpu implementation limits the integer multiples and powers of twos available for such representations to a finite set whose size depends on whether single or double precision is used.

2. Division by 10 involves division by a non-power of two, e.g., the prime number five. Therefore, numbers given to the first decimal place, e.g., 0.1, cannot in general be represented exactly, multiples of 1.0 and 0.5 are exceptions. The representation of most of the numbers therefore will be in error by a fraction of a bit. The sum of numbers given to the first decimal place therefore, cannot, in general, be done exactly in either single or double precision.

3. It is possible that depending. either on the average effects of rounding and on the order of calculations, e.g., whether the sum goes from smallest to largest, or vice versa, the result might accidently be the "correct" answer. On average on such a simple problem, I suspect that single and double precision will loose the same number of bits, but because the double precision has more initially significant bits the relative error in double precision will usually be much smaller than in single precision.

4. It is not generally possible for a compiler or interpreter to know the order of operations which minimizes the error, that requires a knowlege about the detailed properties of the specific set of numbers subject to the operations. It is sometimes possible to use your specific knowlege of the numbers' properties to force the compiler to perform operations in a specific order that minimizes errors, e.g., add the numbers in the order smallest magnitude to largest magnitude to minimize roundoffs, or if the number come in pairs with opposite signs, add the pairs together first. Both methods have runtime performance hits.

5. The conversion in almost any implementation of C, Fortran, or IDL, from the base two internal representation to the base ten output, typically involves rounding, optimally to within the greater of the least significant bit of the representation or the least significant digit of the output, although that is quality of implementation dependent. This rounding is an additional source of error that might or might not result in the "correct" answer. For free format the least significant bit is comparable in magnitude to the least significant digit in a high quality implementation. For the example proplem, however, this is unlikely to be

a significant source of error in the output, as the higher order bits are subtractted out before the output is generated.

--

William B. Clodius Phone (505) 665-9370
Los Alamos Natl. Lab. NIS-1 FAX (505) 665-7395
PO Box 1663, MS-D466 Group Office (505) 667-2701