Technical Note - Double-Rounding and Implications for Numeric Computations

by Cleve Moler

Here is a test of floating-point rounding:

```
eps  = the distance from 1.0 to the
       next larger floating-point number,
     = 2^(-52) for IEEE long arithmetic
e    = a small multiple of eps
Compute ((1+e)*(1.5+e)-1.5)/e
```

With exact computation, the result would be `2.5 + e`, so with roundoff error one might expect to get `2.5`.

Here is a MATLAB code segment that runs this test with three different values of `e`:

```
format  compact
format  hex
e = [0.5 1 2]*eps
a = 1 + e
b = 1.5 + e
c = a.*b
d = c - 1.5
format  short
d ./ e
```

The chart at right shows the results on three different machines.

Ardent Titan:

```
e = [3ca0000000000000  3cb0000000000000   3cc0000000000000]
a = [3ff0000000000000  3ff0000000000001   3ff0000000000002]
b = [3ff8000000000000  3ff8000000000001   3ff8000000000002]
c = [3ff8000000000000  3ff8000000000003   3ff8000000000005]
d = [0000000000000000  3cc8000000000000   3cd4000000000000]
ans = [ 0    3.0000    2.5000 ]
```

Sun-3:

```
e = [3ca0000000000000  3cb0000000000000   3cc0000000000000]
a = [3ff0000000000000  3ff0000000000001   3ff0000000000002]
b = [3ff8000000000000  3ff8000000000001   3ff8000000000002]
c = [3ff8000000000000  3ff8000000000002   3ff8000000000005]
d = [0000000000000000  3cc0000000000000   3cd4000000000000]
ans = [ 0    2.0000    2.5000 ]
```

PC compatible:

```
e = [3ca0000000000000  3cb0000000000000   3cc0000000000000]
a = [3ff0000000000000  3ff0000000000001   3ff0000000000002]
b = [3ff8000000000000  3ff8000000000001   3ff8000000000002]
c = [3ff8000000000000  3ff8000000000002   3ff8000000000005]
d = [0000000000000000  3cc0000000000000   3cd4000000000000]
ans = [ 0    2.0000    2.5000 ]
```

When `e = 0.5*eps`, we see on all three machines that it gets lost when added to `1.0` or `1.5`, so the final result is zero. This would happen for any value of `e` less than `eps`.

When `e = 2.0*eps`, we see that each machine produces the expected final result of `2.5`. This should happen on any reasonable machine for any value of `e` greater than a few `eps`.

The interesting case is when `e` is exactly equal to `eps`. Then `1+e` and `1.5+e` can be expressed as floating-point numbers, so there is no roundoff error involved in computing `a` or `b`. In fact, the only roundoff occurs in computing the product, `c = a.*b`. With exact computation `c` would be `1.5 + 2.5*e + e^2`, which is almost exactly halfway between the two floating-point numbers `1.5 + 2*eps` and `1.5 + 3*eps`. But the `e^2` term makes the exact result a little closer to `1.5 + 3*eps`, so that is what "perfect" rounding should produce. There are no further roundoff errors, and so the final result "should be" `3` instead of `2.5`.

This is what happens on the Titan, which has IEEE floating-point format and follows IEEE rounding rules, but which does not have some of the fancier features of the IEEE standard and its optional extensions.

What about the Sun-3 and the PC when `e = eps`? They both produce `2.0`, rather than `2.5` or `3.0`. Why? The Sun-3 uses the Motorola floating-point chip and the PC uses the Intel floating-point chip. Both of these chips adhere faithfully to the IEEE floating-point standard, including the optional extended precision. The crucial intermediate quantity `1.5 + 2.5*eps` can be represented in extended precision, and it appears that both of these machines do so. A second rounding is required to produce a long precision number, and now the IEEE round-to-even rule leads to `1.5 + 2*eps`, rather than the "correct" `1.5 + 3*eps`.

This phenomenon is known as "double-rounding". It does not occur on machines like the Ardent Titan, that are intended to have fast IEEE floating-point and so do not have extended precision. The Sun Fortran compiler, f77, has many "float options", some of which affect the handling of extended precision. I am not sure which options would change the behavior of a Fortran version of this program. I am even less sure about what would happen with C compilers on the Sun-3, or with either language on a SPARCstation or other machine. ◆

Cleve Moler is co-founder and chairman of The MathWorks, Inc.