Additional Floating-Point Indoctrination Exercises

David Hough

Revised July 21, 1988

I remember the following exercises from my graduate and early industrial days. Those of you who find the exercises set so far by Prof. Kahan to be too few, too difficult, or too easy, may find something more to your liking below. As in real life, problem statements may be incomplete or contradictory; part of the problem is to sort that out. This document is itself an exercise, being the first set by me using LaTeX.

The exercises in Knuth's Chapter 4 contain clues about some of the following; so do some of the other handouts.

1 Floating-Point Implementation

1.1 Ambiguous Case

Consider an IEEE 854 single-precision-only implementation in its default round-to-nearest mode, with p significant digits of radix β. A normalized unrounded digit string representing an infinite-precision value looks like

$$\underbrace{\boxed{\eta??????\lambda}}_{p \text{ to retain}} \underbrace{R?? \cdots}_{\text{discard}}$$

with $\eta > 0$.

The boxed digits represent those that are to be retained in the final result; R is the Round digit; additional digits may be temporarily preserved to the right of the Round digit, but the rightmost must be a Sticky bit or digit that reflects whether additional non-zero digits have already been discarded. The Ambiguous Case arises whenever $R = \delta \equiv \beta/2$ and all digits to the right are zero. It's of interest because 854 requires that the Ambiguous Case be rounded to even, which requires consulting the least significant retained digit λ prior to rounding and propagating any carry. But if it is known that the Ambiguous Case can never arise, then rounding may be conveniently accomplished, regardless of

λ, by adding half in the last place retained, which is the same as adding δ in the Round digit.

Ignoring underflow, for which operations $\in \{+, -, *, /, sqrt\}$ and radices $\beta \in \{2, 10\}$ can the Ambiguous Case never arise? Generalize to consider underflow. Generalize to arbitrary integer radices $\beta > 2$.

1.2 Rounding Carry Out

Consider a normalized unrounded result as before:

$$\boxed{\rho\rho\rho\rho\rho\rho\rho\rho}\ R?? \ldots$$

but this time with $R \geq \delta$ and all retained digits $\rho \equiv \beta - 1$. Whether or not we are in the ambiguous case, the correctly rounded result will be

$$\boxed{10000000}$$

with the exponent adjusted to account for the carry out. We would of course prefer not to have to check for carry out if it can't arise.

Ignoring underflow, for which operations $\in \{+, -, *, /, sqrt\}$ and radices $\beta \in \{2, 10\}$ can Rounding Carry Out never arise? Generalize as in the previous exercise.

1.3 Modulus and Remainder

The names Remainder and Modulus are often applied to certain functions denoted generically as *rem* having definitions like the following for finite x and finite non-zero y:

$$rem(x, y) = x - y * [x/y]$$

where $[x/y]$ is defined to be the nearest integral value to x/y according to some rule. "Integral value" is used in the sense of a mathematical integer of unlimited magnitude, to emphasize that it need not be representable in an integer or even in a floating-point format in a particular computer. The rule by which x/y is rounded to an integral value varies, but by any rounding method

$$|x/y - [x/y]| < 1$$

and among the points $x - y * n$ for integral n, $rem(x, y)$ must be one of the two closest to x. Thus we conclude that $|rem(x, y)| \leq |x|$ and $|rem(x, y)| < |y|$. **Prove these statements.**

Where definitions of *rem* functions differ is in their rules for rounding x/y to an integral value. Using the symbols *mod* for the typical mathematician's "modulo", *arem* for the typical arithmetic "remainder", and *ieee* for the remainder function specified in IEEE 854, the associated definitions and properties are these:

mod(x, y): $[x/y]$ is defined by rounding x/y to the nearest integral value toward $-\infty$, so that $0 \leq x/y - [x/y] < 1$, *mod* lies between 0 and y, and therefore *mod* has the same sign as y.

arem(x, y): $[x/y]$ is defined by rounding x/y to the nearest integral value toward 0, so that $|x/y| - |[x/y]| < 1$, and *arem* has the same sign as x.

ieee(x, y): $[x/y]$ is defined by rounding x/y to the nearest integral value, and to the even one in the Ambiguous Case, so that $-1/2 \leq x/y - [x/y] \leq 1/2$, and the sign of *ieee* need agree neither with the sign of x nor with the sign of y.

Which of these three is exact? How can the exact ones be computed for any x or y? Starting from known values for *mod*, *arem*, or *ieee*, how can you readily compute the other functions?

2 Error Analysis

2.1 Complex Division

Develop a method for computing complex quotients that avoids gratuitous overflow; if necessary look up "Smith, Robert Leroy" in the index to Knuth's volume 2. Suppose that the complex and real parts of the numerator and denominator are normalized numbers. **How does the presence or absence of subnormal numbers affect the error bound on the result?**

2.2 Loss of Significance in Trigonometric Functions

The `TRIG(BA_LIB)` man page in the SVID says

> Both `sin` and `cos` lose accuracy when their argument is far from zero. For arguments sufficiently large, these functions return zero when there would otherwise be a complete loss of significance. In this case a message indicating `TLOSS` error is printed on the standard error output. For less extreme arguments causing partial loss of significance, a `PLOSS` error is generated but no message is printed.

Perhaps ATT was influenced by Cody and Waite:

> Clearly there is a general erosion of the precision of `SIN(X)` for smaller `|X|` despite the careful argument reduction just outlined. The algorithm presented here suggests an error return before the quantum loss of significance in `f` associated with the threshold on `N` mentioned above.

What is the difference between normal roundoff, partial loss of significance, and total loss of significance? What mathematical properties of the *sin* function do these reflect? Prove that any numerical approximation SIN must suffer such catastrophic losses of significance, and estimate how large the argument to SIN must be before these occur.

2.3 Error Analysis of Polynomial Evaluation on a Cray

Compute, as accurately as your equipment permits, the positive zero of $f(x) \equiv x^m + x^{m-1} - 1$ for $m = 2, 3$, and 10. Give error bounds on your answers.

You will want to experiment with at least two different methods for computing error bounds. Start with the ideas in Lecture Notes 6b, in which you want to compute

$$P_j \equiv z \cdot P_{j-1} + a_j$$

but instead, on reasonable machines you get a result that can be expressed

$$p_j \equiv (z \cdot p_{j-1} \cdot (1 - \zeta_{j-1}) + a_j) \cdot (1 - \pi_{j-1})$$

or can just as easily be expressed

$$p_j \equiv (z \cdot p_{j-1} \cdot (1 - \zeta_{j-1}) + a_j)/(1 + \tilde{\pi}_{j-1})\,;$$

the corresponding formulas on CDC and Cray machines are more complicated:

$$p_j \equiv z \cdot p_{j-1} \cdot (1 - \zeta_{j-1}) \cdot (1 - \lambda_{j-1}) + a_j \cdot (1 - \pi_{j-1})$$

or

$$p_j \equiv z \cdot p_{j-1} \cdot (1 - \zeta_{j-1}) \cdot (1 - \lambda_{j-1}) + a_j/(1 + \tilde{\pi}_{j-1})\,.$$

All the Greek letters are bounded by the roundoff level: $| \{\zeta_j, \pi_j, \tilde{\pi}_j, \lambda_j\} | \leq \varepsilon$; and on CDC and Cray, it's no loss to simplify the analysis by assuming $0 \leq \{\zeta_j, \pi_j, \tilde{\pi}_j, \lambda_j\} \leq \varepsilon$.

Derive recurrences for computing bounds on $| P_m - p_m | / \varepsilon$, based on each of these possibilities. The first formula in each pair is more conventional than the second; **which leads to better error bounds?** Study that question experimentally and analytically on both reasonable machines and Crays.

2.4 Wilkinson's Growth Factor g

In his classic, *Rounding Errors in Algebraic Processes* , Wilkinson analyzes the Gaussian Elimination process to find that the attempt to solve $Ax = b$ results in a computed solution x that satisfies $(A + K)x = b$. On page 108 he gives an error bound for K as well as a bound on the residual $\| b - Ax \|$. Both bounds involve a factor g which is a bound on the largest element encountered in any of the reduced matrices during the determination of the factors $LU = A$.

Is this g necessary? Can satisfactory error bounds be produced which depend only on the dimension n and the roundoff level?

3 Confusion Rampant

3.1 Testing Trigonometric Functions

FCVS is a Fortran test suite designed by the Federal Government. The following fragment is from test 12 of FCVS program 820, intended to test complex cosine:

```
      COMPLEX AVC, BVC
      REAL R2E(2)
      EQUIVALENCE (AVC, R2E)

      AVC = CCOS(( 3.1416, 0.0) * (-10000.0, 0.0))
      IF (R2E(1) - 0.99725E+00) fail, 40122, 40121
40121 IF (R2E(1) - 0.99736E+00) 40122, 40122, fail
40122 IF (R2E(2) + 0.50000E-04) fail, pass, 40120
40120 IF (R2E(2) - 0.50000E-04) pass, pass, fail
```

What determines whether an implementation will pass or fail this test?

The Government has since disabled this particular test. **What should be done instead?**

3.2 Iterative Improvement Doesn't Work

Iterative improvement upon the residual is a key technique for improving accuracy of computed results. It's an underlying principle of the Kulisch and Miranker paradigm as well as more traditional error analysis techniques.

However about 15 years ago a prominent Mathematician submitted a paper complaining that iterative improvement didn't seem to help much on realistic problems. He had tested iterative improvement on a number of examples from a book of test matrices for linear algebra software, and had supplemented these with a number of random test matrices. In all cases he found that iterative improvement might improve the computed result by one or two significant digits at most and then didn't seem to do any further good.

Was the learned Professor correct? If so why is the popular theory misleading? If not what was his mistake?