Math. 128 B Four Running Error Bounds 18 Feb. 1997

Prof. W. Kahan, Math. Dept. Univ. of Calif. at Berkeley

Ideally, calculations with rounded arithmetic and uncertain data should be augmented by auxiliary calculations designed to account for errors due to rounding and uncertainty inherited from data by final results. The simplest schemes use Interval Arithmetic, an idea based upon the allocation to every variable of an interval known to contain the variable's value. Then arithmetic operations upon variables are replaced by the same operations upon intervals (actually, only endpoints of intervals need be manipulated), whence results stated as intervals assuredly enclose the desired results. Used naively, Interval Arithmetic can deliver awfully pessimistic (excessively wide) final intervals; to obtain better (narrower but still encompassing) intervals requires considerable skill. Alas, that is all entirely academic for the present because hardly any North American computer systems offer their users access to Interval Arithmetic. The idea is far more popular in Germany than in the USA despite having been devised here in the early 1960s. It's a familiar story; St. Matthew 14:57 ends some parables with the quotation " A prophet is not without honor, save in his own country, and in his own house."

This note describes, for four simple computations, how to manage after a fashion without Interval Arithmetic. Uncertainties in final results are computed by tracking errors and uncertainties through a computation, operation by operation, in a simple but tedious way. Let $\hat{\circ}$ denote a typical operation + , -, * , / .

First an estimate of the arithmetic's rounding error threshold ε is needed; we assume every *assignment* x := y$\hat{\circ}$z actually stores a result $x = (y\hat{\circ}z)/(1 - \alpha)$ where α is unknown but $\varepsilon > |\alpha|$ is known. On correctly rounding machines, (over-)estimate ε thus:

$$\varepsilon := | ((4.0/3.0 \text{ rounded }) - 1.0)*3.0 - 1.0 | .$$

Can you see why this should work? Try it with a calculator first.

Second, we assume that uncertainties in data are known. If we wish to compute X := Y$\hat{\circ}$Z but we know only y and z and error-bounds $eY > |Y-y|/\varepsilon$ and $eZ > |Z-z|/\varepsilon$, then we shall derive another error-bound $eX > |X-x|/\varepsilon$ from x, y, z, eY, eZ and ε . When $\hat{\circ}$ is + then $x = (y+z)/(1-\alpha)$ and eX := eY + eZ + |x| . But eX must be more complicated when $\hat{\circ}$ is * or / .

Consider first a sum $S_N := \Sigma_1^N B_j$ for which we have approximate data b_j and error-bounds $eB_j > |B_j-b_j|/\varepsilon$. We would compute $s_0 := 0$ and $s_N := s_{N-1} + b_N$ but get $s_N = (s_{N-1} + b_N)/(1 - \alpha_N)$, from which we infer $S_N - s_N = S_{N-1} - s_{N-1} + B_N - b_N - \alpha_N s_N$, and then $|S_N - s_N|/\varepsilon <$ $eS_N := eS_{N-1} + eB_N + |s_N|$. We add this last assignment to the program that computes s_N to compute eS_N too, which is our *Running Error-Bound*. Roundoff in eS_N is ignored because it cannot materially affect a sum of positive terms unless N is huge, comparable with $1/\varepsilon$. Here is the complete program:

$s_0 := 0$; $eS_0 := 0$;
for N = 1, 2, 3, ... in turn, do
$s_N := s_{N-1} + b_N$;
$eS_N := eS_{N-1} + eB_N + |s_N|$.

In other words, we augment the recurrence programmed for S_N, but actually carried out upon s_N, with a recurrence carried out simultaneously upon eS_N to get a *Running Error Bound* $eS_N\varepsilon$ for the absolute error $|S_N - s_N|$, ignoring roundoff in eS_N.

The next example computation is a scalar product $S_N := \Sigma_1^N B_j C_j$ for which the data given is b_j, c_j, $eB_j > |B_j - b_j|/\varepsilon$ and $eC_j > |C_j - c_j|/\varepsilon$, and we seek to compute an approximate sum s_N and a bound $eS_N\,\varepsilon$ for its absolute error. Here is what to do:

$s_0 := 0$; $eS_0 := 0$;
for $N = 1, 2, 3, \ldots$ in turn, do
$p_N := c_N * b_N$; $s_N := s_{N-1} + p_N$;
$eS_N := eS_{N-1} + |p_N| + |s_N| + |c_N| * eB_N + (|b_N| + 3\varepsilon * eB_N) * eC_N$.

Now $eS_N > |S_N - s_N|/\varepsilon$, including the effects of roundoff in
$p_N = c_N\, b_N/(1 + \beta_N)$ and
$s_N = (s_{N-1} + p_N)/(1 - \alpha_N)$, where
$|\beta_N| < \varepsilon$ and $|\alpha_N| < \varepsilon$,
but ignoring roundoff in the computation of eS_N itself. Note that subscripted variables (arrays) would not normally be used for p_N, s_N and eS_N in practice.

The next example is a polynomial
$P_N(X) := A_0X^N + A_1X^{N-1} + \ldots + A_{N-1}X + A_N$.
computed from a recurrence
$P_N := A_N + X\,P_{N-1}$ starting with $P_0 := A_0$.
With the same notational conventions as before, the augmented recurrence for p_N and eP_N becomes

$p_0 := a_0$; $eP_0 := eA_0$;
for $N = 1, 2, 3, \ldots$ in turn, do
$t := x * p_{N-1}$; $p_N := a_N + t$;
$eP_N := eA_N + eX * |p_{N-1}| + |t| + |p_N| + (|x| + \varepsilon * eX) * eP_{N-1}$.

At the end, p_N approximates P_N and $eP_N > |P_N - p_N|/\varepsilon$. Later another simpler way will be presented, together with a recurrence and error-bound for the polynomial's derivative, all applied to the problem of finding zeros of a polynomial.

The final example is a continued fraction
$F_0 := A_0 + B_0/(A_1 + B_1/(A_2 + B_2/(A_3 + \ldots + B_M/F_{M+1} \ldots)))$.
The augmented recurrence is valid only if no denominator f_{N+1} is ever smaller than its uncertainty $eF_{N+1}\varepsilon$. Otherwise an entirely different scheme, too complicated to describe here, must be used to account for the possibility that $F_N = \infty$ but F_{N-1} is finite.

For $N = M, M-1, \ldots, 3, 2, 1, 0$ in turn, do
$q := b_N/f_{N+1}$; $f_N := a_N + q$;
$eF_N := eA_N + |q| + |f_N|$
$+ (|b_N| * eF_{N+1} + eB_N * |f_{N+1}|)/(|f_{N+1}| * (|f_{N+1}| - eF_{N+1}\varepsilon))$.

At the end, $eF_0 > |F_0 - f_0|/\varepsilon$.