Applied and Computational Mathematics Division

NISTIR 5724

Computing and Applied Mathematics Laboratory

Error-Bounding in Level-Index Computer Arithmetic

Daniel W. Lozier
Peter R. Turner

October 1995

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards and Technology
Gaithersburg, MD 20899

NIST

NISTIR 5724

Error-Bounding in Level-Index Computer Arithmetic

Daniel W. Lozier
Peter R. Turner

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Applied and Computational Mathematics Division
Computing and Applied Mathematics Laboratory
Gaithersburg, MD 20899

October 1995

PREPRINT

This paper was presented at the International IMACS-GAMM Symposium on Numerical Methods and Error-Bounds, held at the University of Oldenburg, Germany, July 9–12, 1995, and it has been submitted to the Proceedings of the Symposium which will be published as a special volume by Akademie Verlag, Berlin. The paper is subject to revision for compliance with the recommendations and requirements of referees and editors.

Error-Bounding in Level-Index Computer Arithmetic

D. W. Lozier and P. R. Turner

Applied and Computational Mathematics Division
National Institute of Standards and Technology
Gaithersburg, MD 20899

Mathematics Department
United States Naval Academy
Annapolis, MD 21402

Abstract. This paper proposes the use of level-index (LI) and symmetric level-index (SLI) computer arithmetic for practical computation with error bounds. Comparisons are made with floating-point and several advantages are identified.

1 Introduction

Any approach to the general problem of assessing the total error in the output of computer programs depends on a detailed understanding of the computer arithmetic. The finite precision of the arithmetic gives rise to rounding errors that can be an important component of the total error. Accordingly, much effort has gone into refining the algorithms and circuitry that carry out floating-point arithmetic. One goal of this effort has been to minimize rounding errors. Another was to ensure that exceptional conditions, such as underflow and overflow, are detected and reported because their occurrence can completely invalidate the results of a computation. The present state of floating-point hardware design [5] is close to optimal, and so the question arises: Is there a radically different system of arithmetic with properties that are superior?

An answer, proposed a little more than ten years ago [1] and known as level-index arithmetic, is based on representing positive numbers by generalized logarithms. These representations are obtained by repeatedly taking logarithms until a result between zero and unity, the *index*, is obtained. The corresponding *level* is the number of times the logarithm was taken. The level (an integer) and the index (a fraction) are added and stored in a fixed-point location as the internal representation of real positive numbers. This describes the unsymmetric form of level-index arithmetic. There is also a symmetric form [3] in which, effectively, real numbers less than unity in magnitude are reciprocated before being stored.

The main purpose of this paper is to compare the new arithmetic against the old, particularly in regard to interval arithmetic and other error-bounding techniques. Among the advantages will be (i) an immunity to extraneous considerations necessitated by underflow and overflow; (ii) a unified error analysis that naturally blends absolute errors, relative errors, and higher-order generalized errors; and (iii) a natural means for increasing precision when needed within an algorithm.

The representation of real numbers in a computer is based on a mapping of the form

$$X \in \mathcal{S} \subseteq \mathcal{R} \longrightarrow x \in \mathcal{T}_w \subset \mathcal{R} \tag{1}$$

where $\mathcal{S}$ and $\mathcal{T}_w$ are subsets of the real numbers. $\mathcal{T}_w$ is a finite subset associated with computer words of length w bits. Elements of $\mathcal{T}_w$ are called *internal numbers*, those of $\mathcal{S}$ *external numbers.* Usually $\mathcal{S}$ is a finite interval such as $(-M, M)$ or (M^{-1}, M).

To be useful in representing external numbers, the mapping should be invertible but of course this is possible only in an approximate sense. Accordingly, suppose that

(2) $$x = \tilde{f}(X)$$

where $\tilde{f}$ is an approximation to a continuous real function f that is invertible with inverse function g. Then we define the *generalized error function*

(3) $$\text{generr}(X, \tilde{X}) = |f(X) - f(\tilde{X})|$$

where

(4) $$\tilde{X} = g(x)$$

is our external approximation to X.

Commonly used representations are fixed-point, logarithmic and floating-point. The fixed-point representation function is the identity. In this case $\tilde{X} = x$ and, assuming $\mathcal{S} = (-1, 1)$, x is obtained by rounding the binary expansion of X to $w - 1$ bits (one bit is needed for the sign of X). The generalized error is just the absolute error, and the inequality

(5) $$\text{generr}(X, \tilde{X}) = |X - \tilde{X}| \leq 2^{-w}$$

is satisfied.

For the binary logarithmic representation on $\mathcal{S} = (M^{-1}, M)$ with $M = 2^{2^m}$, f is the binary logarithm and $x = \tilde{f}(X)$ is $\log_2 X$ rounded to $w - m - 2$ bits. Then

(6) $$\text{generr}(X, \tilde{X}) = |\log_2 X - \log_2 \tilde{X}| \leq 2^{m+1-w}$$

where $\tilde{X} = 2^x$. Since

(7) $$\log_2 X - \log_2 \tilde{X} = \log_2\{1 + (X - \tilde{X})/\tilde{X}\}$$

and similarly for $\log_2 \tilde{X} - \log_2 X$, the generalized error is, to within terms of first order, just the relative error times $1/\ln 2 = 1.44\cdots$. It may be noted in comparison to fixed-point that an additional sign bit is needed and that the representation of zero is special.

Floating-point may be viewed as a modification of the logarithmic representation. Suppose $X \in (M^{-1}, M)$ and write

(8) $$\log_2 X = c(X) + \log_2\{1 + \sigma(X)\}$$

where $c(X)$ is the unique integer determined by the condition $\sigma(X) \in [0, 1)$. Here $\sigma(X)$ is the fractional part of the floating-point significand and $2^{c(X)}$ is the scale factor. The representation function is not so readily expressed as for the logarithmic representation, since the signs of $c(X)$ and $\sigma(X)$ are opposite when $X < 1$. In effect, it is taken as

$f(X) = \sigma(X)$ with separate accounting for $c(X)$. The internal approximation x is $\sigma(X)$ rounded to $w - m - 2$ bits, and in the usual case when $c(X) = c(\tilde{X})$,

$$\text{generr}(X, \tilde{X}) = |\sigma(X) - \sigma(\tilde{X})| \leq 2^{m+1-w}. \tag{9}$$

For IEEE arithmetic in single and double precision, $m = 7$ and $m = 10$, respectively.

2 The Level-Index Representation

The *generalized logarithm* is, by definition, the function

$$\psi(t) = \begin{cases} t & \text{if } 0 \leq t \leq 1, \\ 1 + \psi(\ln t) & \text{if } t \geq 1, \\ -\psi(-t) & \text{if } t < 0. \end{cases} \tag{10}$$

This function is invertible and its inverse function is given by

$$\phi(t) = \begin{cases} t & \text{if } 0 \leq t \leq 1, \\ e^{\phi(t-1)} & \text{if } t \geq 1, \\ -\phi(-t) & \text{if } t < 0. \end{cases} \tag{11}$$

Both ψ and ϕ are strictly increasing, continuous, and continuously differentiable on R. If $t \geq 1$, $\psi(t)$ is obtained by repeatedly taking logarithms until $\ln^{(\ell)} t = \ln \ln \cdots \ln t \in [0, 1)$. Then $\psi(t) = \ell + \ln^{(\ell)} t$. By definition, ℓ is the *level* and $\ln^{(\ell)} t$ is the *index* of t. For the inverse function, if $t = \ell + a \geq 1$ where ℓ is the integer part of t, then $\phi(t) = \exp^{(\ell)}(a)$.

For level-index, or LI, computer arithmetic we take $\mathcal{S} = (-\phi(8), \phi(8))$ and $x = \tilde{\psi}(X)$ where x is obtained by rounding the binary expansion of $\psi(X)$ to $w - 4$ bits. The following table supports this choice of $\mathcal{S}$:

x	$\phi(x)$	x	$\phi(x)$	x	$\phi(x)$
0	0	3	$15.15\cdots$	$4.63\cdots$	$2^{1024} \approx 10^{308}$
1	1	4	$10^{6.58\cdots}$	$4.80\cdots$	$2^{16384} \approx 10^{4932}$
2	$2.72\cdots$	$4.40\cdots$	$2^{128} \approx 10^{38}$	$4.99\cdots$	$2^{5502841} \approx 10^{1656520}$

We see that $x = 4.40$ corresponds approximately to the IEEE standard overflow threshold in single precision ($w = 32$). Overflow thresholds in double precision ($w = 64$) and quadruple precision ($w = 128$) are reached at $x = 4.63$ and $x = 4.80$, respectively. Allocating 3 bits to the level and $w - 4$ to the index (the remaining bit is the sign bit) allows us to represent numbers in the vast interval $(-\phi(8), \phi(8))$. Indeed, $\phi(6)$ is already so large that it is impractical to express it as a floating-point number.

If restricted to the interval $(-1, 1)$, LI arithmetic is equivalent to fixed-point. A symmetric modification, called SLI arithmetic, is more analogous to floating-point.

We take $\mathcal{S} = (M^{-1}, M)$ where $M = \phi(8)$. The representation function becomes

(12) $$\Psi(t) = \psi(\ln t) = \begin{cases} \psi(t) - 1 & \text{if } t \geq 1, \\ 1 - \psi(1/t) & \text{if } 0 < t < 1 \end{cases}$$

with inverse function

(13) $$\Phi(t) = e^{\phi(x)} = \phi(x+1).$$

Again, 3 bits are allocated to the level. With one bit each for the signs of X and $\Psi(X)$, $w - 5$ bits are allocated to the index.

In [6] it is proved that LI and SLI arithmetic with 3 bits allocated to the level are both *closed*. That is, all sums, differences, products, and quotients, excluding division by zero, of numbers in $\mathcal{T}_w$ are elements of $\mathcal{S}$, provided only that w does not exceed 5 million bits or so. Thus the rounded result of an arithmetic operation in $\mathcal{T}_w$ is again in $\mathcal{T}_w$. In particular, this means that both overflow and underflow have been abolished for the basic arithmetic operations.

The generalized error for LI and SLI is defined by (3) and (4) with f, g replaced by ψ, ϕ and Ψ, Φ, respectively:

(14) $$\operatorname{generr}(X, \tilde{X}) = \begin{cases} |\psi(X) - \psi(\tilde{X})| \leq 2^{3-w} & \text{for LI,} \\ |\Psi(X) - \Psi(\tilde{X})| \leq 2^{4-w} & \text{for SLI;} \end{cases}$$

cf. (5), (6) and (9). At level 1, when $1 \leq X \leq e$, the generalized error is the relative error in the external approximation (to within terms of first order). The behavior of relative error for $X > e$ is the subject of the next section.

3 Representation Errors

For a given computer arithmetic the generalized error is measured in the set $\mathcal{T}_w$ of internal numbers: it is bounded uniformly by a small constant that depends on w. For fixed-point, logarithmic and floating-point arithmetics the generalized error has a familiar interpretation in the external set $\mathcal{S}$: the number of either 'decimal places' (in fixed-point) or 'significant decimal digits' (in logarithmic and floating-point) is uniformly bounded. There is no familiar interpretation of generalized error for level-index arithmetic. Accordingly, a comparison in familiar terms is needed.

Figure 1 presents a comparison of SLI against IEEE floating-point for $w = 32$ and $\mathcal{S} = [1, 10^{45}]$. The horizontal scale is $\log_{10} X$ for $X \in \mathcal{S}$. The vertical scale, $\mu(X)$, is a measure of 'significant decimal digits' computed by evaluating the formula

(15) $$\mu(X) = -\log_{10} \frac{X^+ - X}{X}$$

in double precision. Here the equations

(16) $$\Psi(X^+) = \Psi(X) + 2^{-27}, \qquad \sigma(X^+) = \sigma(X) + 2^{-23}$$

determine X^+ and X.

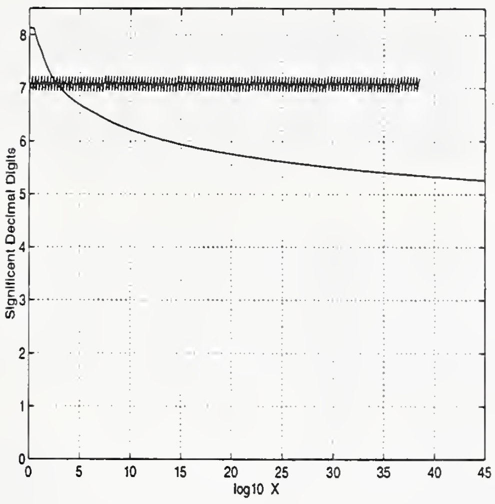

FIGURE 1. Comparison of SLI against IEEE for $w = 32$.

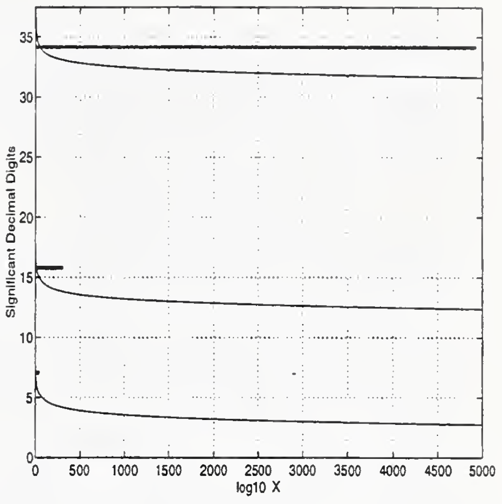

FIGURE 2. Comparison of SLI against IEEE for $w = 32, 64, 128$.

Figure 1 illustrates differences between 32-bit IEEE and SLI arithmetic. First, the IEEE curve exhibits oscillatory behavior not present in SLI. This is due to the phenomenon [4][1] known as *wobbling precision.* The logarithmic curve, were it shown, would lie within the oscillatory band and would be essentially constant. The SLI curve is smooth and gradually decreasing. Second, the IEEE curve does not extend beyond the overflow limit of 10^{38}, approximately. Numbers beyond this limit have no IEEE representation other than a generic infinity symbol, whereas SLI retains useful significance far beyond the limit. Third, the two curves cross at approximately $X = 2400$. Before this point SLI has more significance, while beyond it IEEE does until it fails at the overflow limit.

Figure 2 compares 32-, 64- and 128-bit IEEE[2] and SLI over the range $\mathcal{S} = [1, 10^{5000}]$. In the IEEE formats the overflow limit is increased as w increases by extending the width of the field that holds $c(X)$; cf. (8). The field widths are 8, 11 and 15 bits, respectively. Accordingly, as w increases the SLI index field gains more bits than the IEEE significand field. This results in the crossover point increasing from 2400 in the 32-bit format to approximately 10^{13} and 10^{97} in the 64- and 128-bit formats, respectively. In computing applications that involve numbers that lie mostly to the left of the crossover point, the relative precision of SLI should exceed IEEE.

In the authors' experience instances are known where double precision is used in practical computations not because single precision is too inaccurate but to avoid

[1]The phenomenon occurs for radix 2 as well as for higher floating-point radices, contrary to what is claimed in [4].

[2]Strictly, the IEEE standard does not specify 128-bit formats. The format used here is a plausible extension that has been used in commercial computer products.

overflow. In SLI overflow (and underflow) are impossible, so the precision can be chosen solely on accuracy requirements. Another advantage is that the field for the level is always 3 bits. Without a clear mathematical criterion for subdividing the floating-point word into its two constituent subfields, a variety of inconsistent formats has emerged. This is still true even after widespread adoption of the IEEE standard.

4 Concluding Remarks

The active developers of LI and SLI are small in number but they have produced a body of literature on algorithms, applications and error analyses some of which is contained in the 1989 survey [2]. This reference summarizes the recursive algorithms that are used to perform the basic LI and SLI arithmetic operations in fixed-point arithmetic with a small number of guard digits. The 1995 paper [7], which discusses present and planned software simulations, lists additional references in its bibliography.

LI and SLI possess several advantages compared to floating-point. Some of these have been introduced in this paper. Freedom from underflow and overflow is the greatest advantage. Error analysis in terms of generalized errors may appear to be an obstacle but it may have advantages, for example in appropriately measuring computational error in the neighborhood of a zero. This possibility will be taken up in a future paper. Finally, in contrast to floating-point, an increase or decrease in wordlength to accommodate changing needs for precision is achieved naturally in LI and SLI.

References

1. Clenshaw, C. W. and Olver, F. W. J.: Beyond floating point. *J. Assoc. Comput. Mach.* 31 (1984), 319–328.
2. Clenshaw, C. W.; Olver, F. W. J. and Turner, P. R.: Level-index arithmetic: An introductory survey. *Numerical Analysis and Parallel Processing, Lecture Notes in Mathematics 1397 (P. R. Turner, ed).* Berlin: Springer–Verlag 1989, 95–168.
3. Clenshaw, C. W. and Turner, P. R.: The symmetric level-index system. *IMA J. Numer. Anal.* 8 (1988), 517–526.
4. Cody, Jr., W. J. and Waite, W.: *Software Manual for the Elementary Functions.* Englewood Cliffs, New Jersey: Prentice-Hall 1980.
5. IEEE: *IEEE Standard for Binary Floating-Point Arithmetic.* New York: The Institute of Electrical and Electronics Engineers, Inc., 1985.
6. Lozier, D. W. and Olver, F. W. J.: Closure and precision in level-index arithmetic. *SIAM J. Numer. Anal.* 27 (1990), 1295–1304.
7. Lozier, D. W. and Turner, P. R.: Parallel and serial implementations of SLI arithmetic. *NIST Internal Report 5660.* June 1995. Submitted to *Theoretical Computer Science.*