ROM-rounding: A new rounding scheme

Article · November 1975

DOI: 10.1109/ARITH.1975.6156995

3 authors, including:

Ahmed Sameh
Purdue University
308 PUBLICATIONS **5,172** CITATIONS

SEE PROFILE

ROM-ROUNDING: A NEW ROUNDING SCHEME*

D. J. Kuck, D. S. Parker, and A. H. Sameh
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Abstract

ROM-rounding is introduced and is shown to compare favorably with existing floating-point rounding methods on design considerations and on performance over a series of error tests. The error-retarding value of guard digits, of rounding the aligned operand, and of rounding in general are discussed.

1. Introduction

Existing computer arithmetic units use various methods to reduce floating-point numbers to shorter-precision approximations. The standard methods in use are

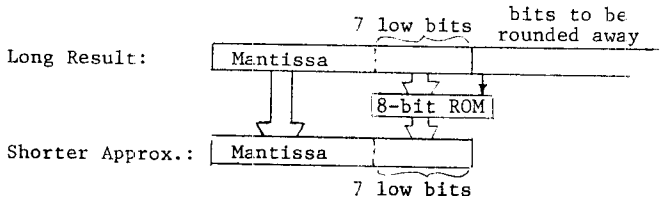
(1) ordinary Truncation (or "chopping").
(2) ordinary Rounding.

Other methods have been proposed for their cost-effectiveness or for their error resistance properties, e.g.,

(3) von Neumann rounding (equivalent to Truncation except that the lowest order bit of the result mantissa is forced to a 1).
(4) R*-rounding (equivalent to Rounding except when the digits to be rounded away have bit balues 10000..., in which case von Neumann rounding is used.)

This paper compares these methods with a fifth method, ROM-rounding, an attractive alternative both from a standpoint of cost and error resistance. With ROM-rounding the (say) 7 low-order bits of the significant part of the mantissa and the highest bit to be rounded away, together form an 8-bit input ("address") to a ROM or PLA. The output of the ROM/PLA is the 7-bit rounded value of these bits, except in the threshold case when all 7 low bits of the significant part of the mantissa are 1--the case in which rounding overflow might occur--rounding is not performed. This is illustrated in Fig. 1.

Figure 1. ROM-rounding



Note that ROM-rounding provides the advantages of rounding (255 times out of 256 for the 8-bit scheme above) without the timing or hardware overhead of the traditional additional rounding adder stage. It is also a cheap scheme to implement in modern technologies. Thus, from an engineering standpoint alone, ROM-rounding is an appealing way to build rounded arithmetic into a floating-point unit. Since its arithmetic properties are sound, it remains to verify only that its error resistance is adequate. The rest of this paper shows that its resistance is not only adequate, but superior: with respect to three indicators of rounding scheme effectiveness, viz.,

(1) Average relative representation error (ARRE)
(2) Rounding scheme bias
(3) Statistical tests with specific problem types

ROM-rounding gives uniformly excellent results. The paper also investigates the value of guard digits (extra low-order digits in floating-point hardware that hold part of the aligned operand, which is shifted right during exponent comparison, giving the final result greater precision; this result may then be rounded to single precision), and the value of alignment rounding (i.e., the use of a rounding method to reduce the aligned operand to fit in the aligned word and guard digits before addition or subtraction). To our knowledge, the latter subject has not previously been discussed--Truncation has simply been used.

2. Representation Error

A theoretical metric of rounding scheme effectiveness is the average relative representation error (ARRE) incurred by the scheme in representing real numbers in floating-point format. McKeeman [5] and later Cody [2] used this metric for comparison between base 2, 4, and 16 arithmetic systems. They defined the ARRE of a t-digit, base β, sign-magnitude system using Rounding by

$$\mathrm{ARRE}(t,\beta) = \int_{1/\beta}^{1} \left(\frac{\beta^{-t}}{4x}\right) \frac{dx}{x \ln\beta} = \left(\frac{\beta^{-t}(\beta-1)}{4 \ln\beta}\right)$$

where $(1/x \ln\beta)$, the reciprocal density, has been used because of the logarithmic law (of leading digits in floating-point computation, see e.g., [8]) and

$$\left(\frac{\beta^{-t}}{4x}\right) = \frac{\text{Average } (|\mathrm{Round}(x,t) - x|)_{1/\beta<x<1}}{x}$$

is used as an approximation to the absolute value of the relative error

$$|\delta(x)| = \left|\frac{\mathrm{Round}(x,t) - x}{x}\right| .$$

With this in mind we redefine, for any rounding scheme,

$$\mathrm{ARRE}(t,\beta) = \int_{1/\beta}^{1} |\delta(x)| \frac{dx}{x \ln\beta} .$$

This quantity was evaluated for each of the five rounding schemes (via a 20,000-point Monte Carlo integration) using $\beta = 16$ and various values of t. The results are plotted in Fig. 2, and agree excellently with the theoretical values for Truncation and Rounding. Using the error density functions for Truncation and Rounding in [8], we expect

$$\mathrm{ARRE}(t,\beta)_{\text{Truncation}} = \frac{\beta^{-t}(\beta-1)}{2 \ln\beta}$$

$$\mathrm{ARRE}(t,\beta)_{\text{Rounding}} = \frac{\beta^{-t-1}(\beta^2-1)}{4 \ln\beta}$$

*This work was supported in part by the National Science Foundation under Grant No. NSF DCR73-07980 A02.

A glance at the plot shows 8-bit ROM-rounding is only marginally worse than regular Rounding for ARRE---this is reasonable not only because the two are identical 255 times out of 256, but also because the one case in which they differ (inputs to the ROM are all 1) has the lowest individual relative error.

3. Rounding Scheme Bias

While ARRE measures the average magnitude of roundoff errors, we need some indicator of a rounding scheme's tendency to favor errors of a particular sign. Consider the set M(t+g) of all normalized binary mantissas of length t+g, which are to be represented in M(t)--i.e., rounded to t bits. We can define

$$\text{Average Bias } (t,g) = \frac{\sum_{x\in M(t+g)} \text{round}(x,t) - x}{||M(t+g)||} .$$

This is not a perfect measure of bias since it evaluates absolute and not relative errors, but it is very informative and makes computation of the following table possible:

Table 1

Scheme for round (x,t)	Average Bias (t,g) with Base 2 arithmetic	
Truncation	$2^{-1-t}(2^{-g} - 1)$	
von Neumann	$2^{-1-t}(2^{-g})$	
Rounding	$2^{-1-t}(2^{-g})$	
R*	0	
ROM (ℓ)	$2^{-1-t}(2^{-g} - 2^{1-\ell})$	(ℓ is the ROM length in bits; $\ell<t+1$)

This table is easily modified to hold for any numeric base $\beta = 2^n$, $n > 1$. Because mantissas in base β have from 0 to n - 1 leading binary zeroes, average bias can be computed by treating all mantissas in base 2 and summing up the different cases:

$$\text{Average Bias } (\tfrac{t}{n},\tfrac{g}{n})_{\text{Base }\beta} = \frac{\sum_{i=0}^{n-1} \text{Average Bias } (t-i,g)_{\text{Base 2}} \; ||M(t-i+g)||}{\sum_{i=0}^{n-1} ||M(t-i+g)||}$$

Now using

$$||M(t-i+g)|| = 2^{t-i+g-1}$$

and

$$\text{Average Bias } (t-i,g)_{\text{Base 2}} = 2^i \cdot \text{Average Bias } (t,g)_{\text{Base 2}}$$

we find

$$\text{Average Bias } (\tfrac{t}{n},\tfrac{g}{n})_{\text{Base }\beta} = \text{Average Bias } (t,g)_{\text{Base 2}} \cdot \left[\frac{\sum_{i=0}^{n-1} 2^i \, 2^{t-i+g-1}}{\sum_{i=0}^{n-1} 2^{t-i+g-1}} \right]$$

$$= \text{Average Bias } (t,g)_{\text{Base 2}} \cdot \left[\frac{\beta \log_2 \beta}{2(\beta-1)} \right] .$$

Thus if all entries in the table are multiplied by $[\beta \log_2\beta/2(\beta-1)]$, and t and g are replaced by t/n and g/n, respectively, the table holds for base β normalized arithmetic.

The ROM-rounding scheme has particularly interesting bias properties. The artificiality of the method in the threshold case causes the factor

$$(2^{-g} - 2^{1-\ell})$$

to appear in its average bias. If we think of the g bits to be rounded away as the guard bits in a floating-point operation (c.f. [3] for a definition of guard bits), then choosing the ROM length ℓ such that $\ell - 1 = g$ will give the ROM scheme zero average bias. Selection of the proper length ROM is vital for good error performance. Fig. 3 shows what happens when different length ROMs are used for mixed-sign summations (see next section for details) with one and two hexadecimal guard digit computation. Note that for all practical purposes ordinary Rounding on a 360-like hexadecimal machine can be thought of as 25-bit ROM-rounding (6 hex digits + 1 rounding bit). Ordinary Rounding would, therefore, have very nice error properties if this machine had six guard digits (24 guard bits). But this is unreasonable, and will presumably be unreasonable on all machines, so ROM-rounding seems preferable to ordinary Rounding for good bias.

It has been proposed that an R* variation on ROM-rounding might be a superior method. There are three arguments against this proposal:

(1) Increased Cost: ALL of the g bits to be rounded away, rather than just the high-order bit, would somehow have to be (multiplexed or otherwise) fed into the ROM.

(2) Increased Bias: The average bias for this ROM/R*-rounding can be shown, as follows, to be $-2^{-1-t}(2^{1-\ell})$ (compare Table 1). From the definition of Average Bias, we have

$$\text{Average Bias [ROM/R*]} = \text{Average Bias [R*]} + \Big(\sum_{x \in M(t+g)} \text{ROM/R*}(x) - \text{R*}(x) \Big) / ||M(t+g)||$$

Now $(\text{ROM/R*}(x) - \text{R*}(x)) = -2^{-t}$ if x is such that the ROM inputs are all ones; otherwise $\text{ROM/R*}(x) = \text{R*}(x)$. Since Average Bias [R*] = 0 and the number of x in M(t+g) with ROM inputs all ones is $2^{t-\ell+g-1}$,

$$\text{Average Bias [ROM/R*]} = 0 + (-2^{-t})(2^{t-\ell+g-1})/(2^{t+g-1})$$

$$= -2^{-1-t}\ (2^{1-\ell})$$

as stated. This cannot be forced to zero as with ROM-rounding.

(3) Questionable Utility: As will be shown in the next section, ROM-rounding gives roundoff performance as good as that of R*-rounding. Need for further improvement must be demonstrated.

4. Discriminating Problems

A number of papers appearing in 1973 ([1], [4], [6]) attempted to discriminate between floating-point systems which used various radices and rounding schemes by applying varied sets of data to a given problem, and statistically measuring the resultant roundoff errors. Except in [1], the problems typically consisted just of repeated arithmetic operations--for our purposes we will only examine repeated addition and/or subtraction because:

(1) Local roundoff errors in multiplication and division are distributed exactly like representation errors and can be measured with ARRE. The theory of accumulated multiplicative errors is trivial (see [9]).

(2) Only with addition and subtraction is R*-rounding really different from regular Rounding, because the special circumstance under which the two are different (the bits to be rounded away have bit values 100000...) is much more likely to occur.

(3) Only with addition and subtraction can the effects of a) rounding the aligned operand, b) guard digits, and c) a) and b) together, be measured.

With this in mind, the error growths were analyzed for three separate problems: repeated serial addition (all summands positive), repeated serial subtraction (intermediate result and subtrahend forced positive before subtraction), and mixed-sign addition (50% of the summands negative). All problems were performed for six-digit hexadecimal arithmetic on an IBM 360/75 over 200 sets of 1024 summands. The summands were randomly generated from a logarithmic distribution (having the reciprocal density; c.f. [8]) and possessed a maximum base 16 exponent difference of 3 (c.f. [7, p. 39]). At several points throughout each computation the hexadecimal significance of the intermediate result--defined here as the base 16 logarithm of its relative error--was calculated. At the end, these significances were averaged over the 200 observations and plotted. Throughout the computations, the ROM length used was chosen so as to give ROM-rounding zero bias, as suggested in section 3. Thus, for one guard digit of 4 bits, a 5-bit ROM was used. Note that some of the plots have two lines drawn for individual rounding methods, corresponding to results for the different alignment rounding schemes which are indicated in parentheses.

As with ARRE and Average Bias, it should be pointed out that the plots in themselves are not complete indicators of rounding scheme performance. First, the curves plotted are averages and say nothing specific about error behavior for a particular set of data. (Although for the 200 sets of summands used the standard deviations of hexadecimal significance were usually less than 0.5--providing a clear discrimination in almost all cases between Truncation and the other methods). Second, the data was chosen to be "commonplace" but obviously most programs will not use identically distributed data. Third, the problems themselves are mildly contrived in that few programs will serially add up thousands of such numbers. The plots serve to indicate the tenacity of the rounding schemes against error, and reflect some of the facets of accumulated error not detectable with metrics of local roundoff error like ARRE and Average Bias.

4.1 Addition

Fig. 4 shows the error growth for repeated addition. The initial bumps result from the loss of significance involved in hexadecimal postnormalization shifting: the relative errors actually decrease for most methods because the magnitude of the sum's mantissa increases faster than the magnitude of accumulated roundoff error (so, the relative error goes down), but when carry-out finally occurs around 256 operations, normalization of the mantissa gives it a leading digit 1 plus loss of the low-order digit and a sudden jump in relative error results.

More marked in addition of positive operands than in the subtraction problems is the interplay of guard digits and rounding of the aligned operand. The fewer guard digits used, the more effect alignment rounding will have. In fact, guard digits are important for addition only in the way they enhance or inhibit alignment rounding. Note that for two guard digit (8 guard bit) addition, alignment rounding has little effect, while for one guard digit it is extremely important. With zero guard digits the rounding schemes are not defined.

The case of one guard digit is interesting. The magnitude of the sum is increased by round-ups, and alignment round-ups if the guard digit has value $\frac{\beta}{2} - 1$; and is decreased by round-downs, alignment round-downs if the guard digit has value $\frac{\beta}{2} - 1$, and carry-outs. Thus the error is extremely sensitive to rounding bias. Rounding with Rounded alignment gets bad results because of amplified bias: for example, computing (12. + .45) in 2 digit, 1 guard digit, base 10 arithmetic would give the answer 13. Because Rounding has positive bias and Truncation negative bias, Rounding with Truncated alignment results in some bias cancellation. It is for this reason that, with Truncated alignment, Rounding performs better on addition than ROM- or R*-rounding. The bias cancellation is, moreover, good enough that Rounding with Truncation competes with the rounded-alignment ROM and R* schemes. At any rate the statement in [4] that R*-rounding is "unbiased" is untrue if one does not use an unbiased alignment rounding scheme.

4.2 Subtraction

We now concentrate on the effects of subtraction on error growth. Figs. 5 and 6 trace the average increase in relative error for, respectively, repeated serial differences and the more realistic problem of the serial summation of numbers roughly half of which are negative. Truncation performs much better for these two problems than for addition because the local errors are not always of the same sign (as they were in addition), resulting in some error cancellation. This is especially noticeable in Fig. 6, where Truncation with 0 guard digits eventually outperforms Rounding with 1 guard digit. This phenomenon was first observed by Kuki and Cody in [4]; they also pointed out that additional guard digits harm Truncation for mixed-sign summations because guard digits inhibit the error cancellation Truncation relies on so heavily. Kuki and Cody correctly claim that use of the unbiased R*-rounding method produces the best results--but they leave the impression that regular Rounding is always worse than Truncation with 0 guard digits for mixed-sign summations, which is true only if one guard digit is used with Rounding and the sum has many operands (at least 300, using Fig. 6). The high error growth rate of Rounding with one guard digit significantly declines when two guard digits are used.

Note that the hexadecimal significance plots for subtraction indicate a gradual significance loss totally different from the graphs of Figs. 10-12 in [6]. The method of [6] is questionable, with its plots of median elements and statistical basis of two observations. However, duplicated simulation showed conclusively that the dramatic error growths in Figs. 10-12 are due exclusively to the rescaling of all numbers into [0.25,4.00), which confessedly "will tend to accentuate the error growth under subtraction, as prospective arguments are brought closer together in value" [6, p. 589]. Although it is not a benign process, floating-point subtraction is not usually malignant.

5. Conclusion

The main conclusion is clear: ROM-rounding is not only technologically one of the cheapest schemes to implement, but also one of the best schemes in guarding against roundoff errors. Not only does it have excellent bias and representation error properties, but it makes feasible the use of alignment rounding. Alignment rounding decreases the bounds on local roundoff errors in addition and subtraction (derived in [3]), although it is apparently useful only when few guard digits are employed. ROM-rounding can probably be used--especially with binary machines having only a couple of guard bits--to exploit the tradeoff between guard bits and alignment rounding, producing an extremely cost-effective rounding scheme.

References

[1] R. P. Brent, "On the precision attainable with various floating-point number systems," IEEE Trans. Comput., vol. C-22, no. 6, p. 601-607, June 1973.

[2] W. J. Cody, "Static and dynamic numerical characteristics of floating-point arithmetic," IEEE Trans. Comput., vol. C-22, no. 6, pp. 598-601, June 1973.

[3] T. Kaneko and B. Liu, "On local roundoff errors in floating-point arithmetic," J. ACM, vol. 20, no. 3, pp. 391-398, July 1973.

[4] H. Kuki and W. J. Cody, "A statistical study of the accuracy of floating-point number systems," Comm. ACM, vol. 16, no. 4, pp. 223-230, April 1973.

[5] W. M. McKeeman, "Representation error for real numbers in binary computer arithmetic," IEEE Trans. Comput., vol. EC-16, pp. 682-683, Oct. 1967.

[6] J. D. Marasa and D. W. Matula, "A simulative study of correlated error propagation in various finite-precision arithmetics," IEEE Trans. Comput., vol. C-22, no. 6, pp. 587-597, June 1973.

[7] T. Sweeney, "An analysis of floating-point addition," IBM Syst. J., vol. 4, no. 1, pp. 31-42, 1965.

[8] N. Tsao, "On the distributions of significant digits and roundoff errors," Comm. ACM, vol. 17, no. 5, pp. 269-271, May 1974.

[9] J. H. Wilkinson, Rounding Errors in Algebraic Processes. Englewood Cliffs, N. J.: Prentice-Hall, 1963.

Figure 2. ARRE for different schemes

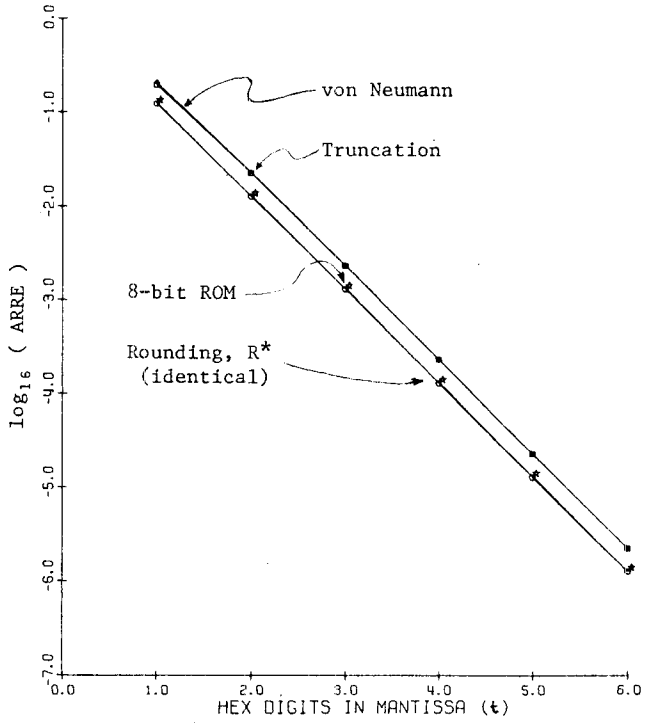


Figure 3. Effect of ROM length on error growth

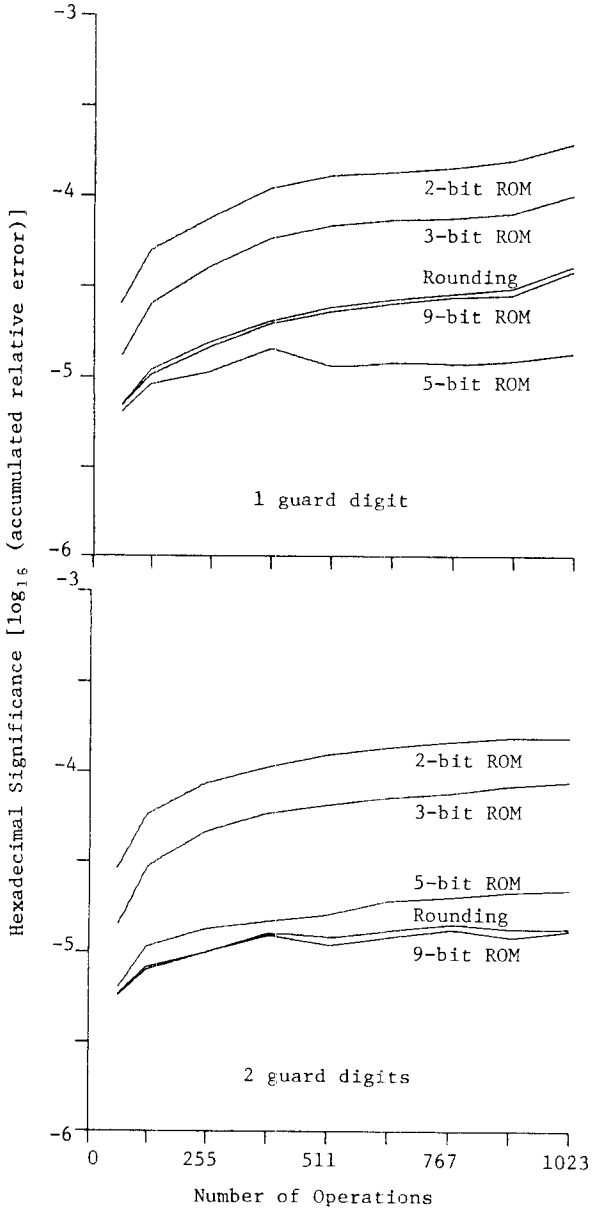
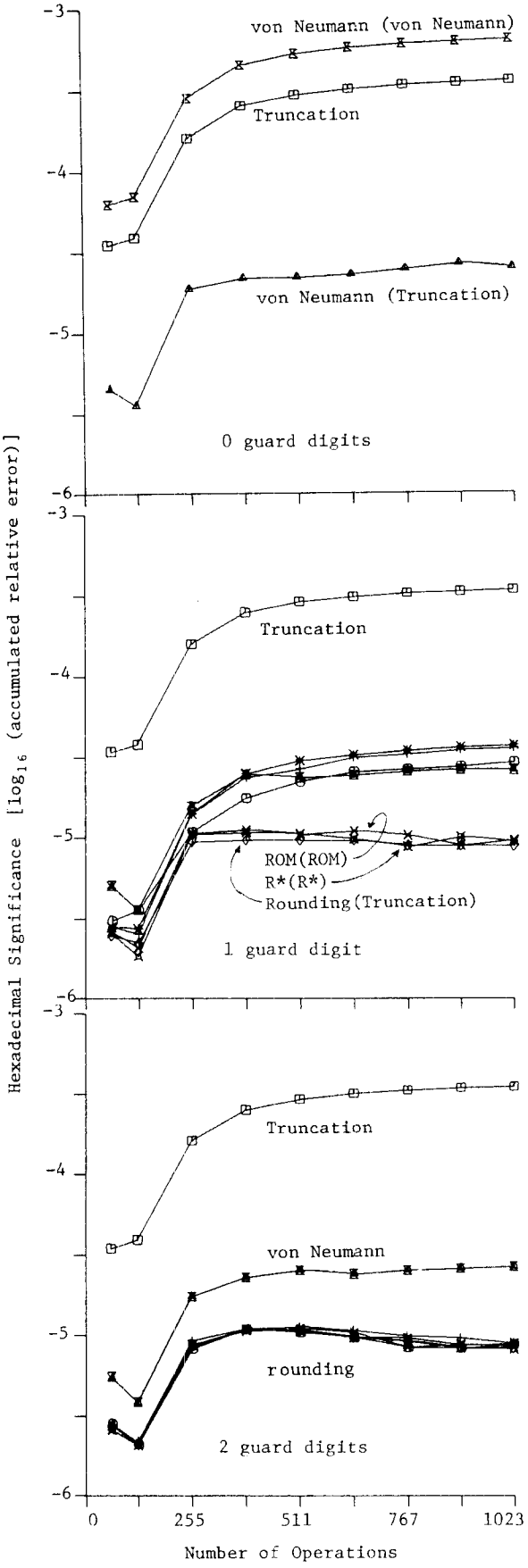


Figure 4. Serial addition error growth



Legend for Figures 4-6

Symbol	Rounding Scheme	(Alignment Scheme)
□	Truncation	Truncation
△	von Neumann	Truncation
⧖	von Neumann	von Neumann
◇	Rounding	Truncation
○	Rounding	Rounding
+	ROM-rounding	Truncation
×	ROM-rounding	ROM-rounding
✳	R*-rounding	Truncation
⋈	R*-rounding	R*-rounding

Figure 5. Serial subtraction error growth

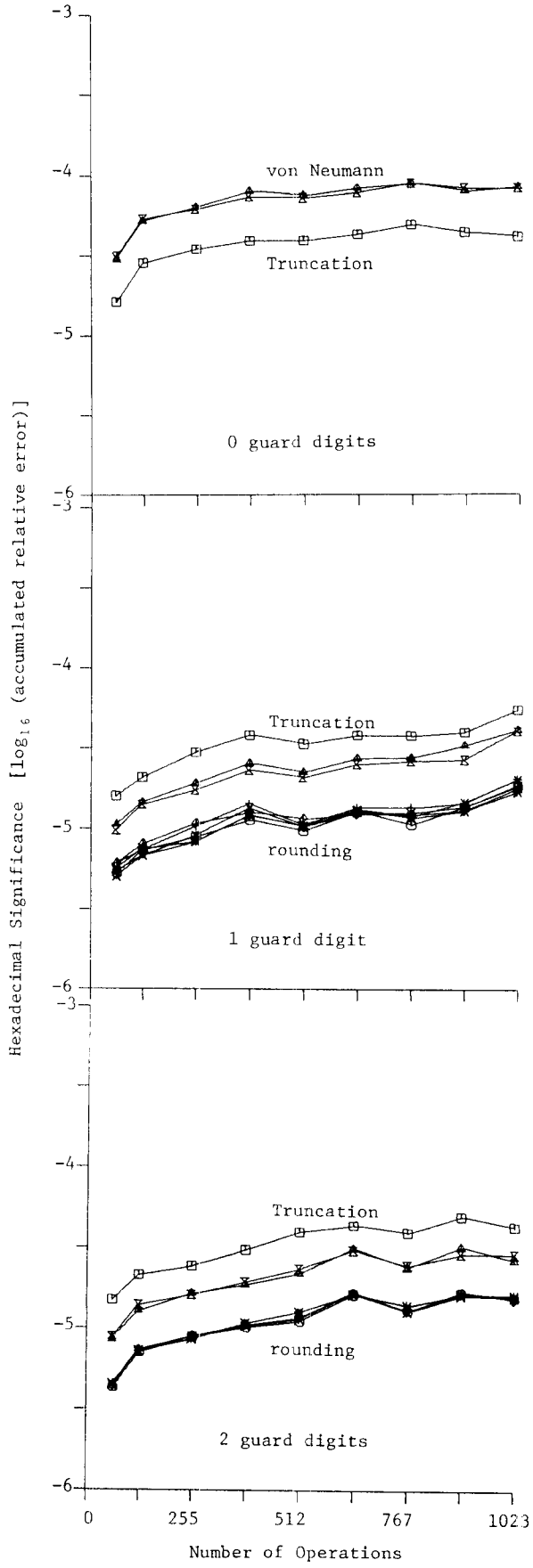


Figure 6. Mixed-sign addition error growth