Optimal Parameters for On-Line Arithmetic

Colin D. Walter

Computation Department, U.M.I.S.T.,
PO Box 88, Sackville Street, Manchester M60 1QD, U.K.
e-mail: C.Walter@umist.ac.uk

Abstract. Some general techniques are given for constructing divergent examples for on-line arithmetic operations whose parameters are chosen beyond the optimal ones which converge. They are applied in particular to the cases of fully on-line multiplication, division and square root.

Index Terms: Computer arithmetic, multiplication, division, square root, on-line algorithms, redundant number systems, recurrence relations.

1 Introduction

The choice of best parameters in hardware algorithms often depends on the use of continuous mathematics to deduce limiting situations, whereas parameters which must fail can be deduced by the application of information theoretic methods. With algorithms that gain speed by performing only partial or approximate calculations at each step there is usually a gap between such sets of parameters, where behaviour is unknown. The discrete nature of hardware arithmetic suggests that it may be impossible for the limiting cases used in the continuous arguments to arise in practice. This would mean that better parameters might be possible. Explicit counter-examples are then required to justify rejecting some parameters choices in this gap, and more detailed, specific proofs for accepting other choices. We develop some techniques for this, considering the cases of fully on-line algorithms for multiplication, division and square root where the iterative step is as simple as possible to gain speed.

Fully on-line algorithms consume all their inputs and generate all outputs digit serially at the same rate, most significant bit first. Multiplication, division and square root have been extensively studied in the literature, using either a simple addition for the iterative step (see [4], [7], [8], [2], [5]), or a fuller, more accurate computation (see [1], [6]), and sometimes with not all inputs being on-line (see [3], [9]). However, a redundant number system must always be used to enable digit parallel addition, and to enable some freedom in the choice of the output digits. There is a natural, fixed time delay δ between input and output of corresponding digits: the output digit of index j depends solely on the input digits of index $j+\delta$ or less. Here the output digit depends on an approximation, down to digits of index ε, of a *partial remainder*, which is basically the difference between the function on the inputs so far and the output so far. The best choice of the parameters δ and ε is to minimise them.

2 The Algorithms

Uppercase characters are used to denote real numbers and lowercase characters to denote digits. We consider real inputs and outputs M with $|M| < 1$ and radix 2 representations $M \equiv \sum_{t=1}^{\infty} m_t 2^{-t}$, where the digit m_t lies in the redundant signed digit set $\{-1, 0, +1\}$. At time $j \geq 0$, the first j digits provide the inputs $M[j] \equiv \sum_{t=1}^{j} m_t 2^{-t}$ from which the output digit of index $j-\delta$ is formed. The difference between the function on such inputs and the previous output yields a scaled *residual error* or *partial remainder*, here defined by:

$$\begin{aligned} W[j] &= 2^j\{X[j]\times Y[j] \ - \ P[j-1-\delta]\} \\ W[j] &= 2^j\{X[j] - Q[j-1-\delta]\times Y[j]\} \\ W[j] &= 2^j\{X[j] - S[j-1-\delta]^2\} \end{aligned}$$

for product $P = X\times Y$, quotient $Q = X/Y$ and square root $S = \sqrt{X}$. The output digit of index $j-\delta$ is chosen to approximately minimise this if it were included in the definition. To guarantee outputs in the interval $(-1, +1)$ we require also $|X| < 1/2 \leq |Y|$ for division and $X \geq 1/4$ for square root. The partial remainder definitions give the recurrence relations

$$\begin{aligned} W[j+1] &= 2W[j] \ + \ y_{j+1}X[j] \ + \ x_{j+1}Y[j] \ + \ 2^{-1-j}x_{j+1}y_{j+1} \ - \ 2^{1+\delta}p_{j-\delta} \\ W[j+1] &= 2W[j] \ + \ x_{j+1} \ - \ y_{j+1}Q[j-1-\delta] \ - \ 2^{1+\delta}q_{j-\delta}Y[j+1] \\ W[j+1] &= 2W[j] \ + \ x_{j+1} \ - \ 2^{2+\delta}s_{j-\delta}\times S[j-1-\delta] \ - \ 2^{1+2\delta-j}{s_{j-\delta}}^2 \end{aligned}$$

with initial value $W[0] = 0$. So the output digits are chosen close to $2^{-\delta}W[j]$, $2^{-\delta}W[j]/Y[j]$ and $2^{-1-\delta}/S[j-1-\delta]$ respectively. Speed is gained by considering just $W[j]'$, given by truncating all digits in $W[j]$ after that of index ε. This is compared with a fixed integer i (depending on the algorithm) and the output digit $r_{j-\delta}$ is then defined by

$$\begin{aligned} r_{j-\delta} = +1 &\text{ if } \ 2^{\varepsilon}W[j]' \ \geq \ i \\ 0 &\text{ if } |2^{\varepsilon}W[j]'| < \ i \\ -1 &\text{ if } \ 2^{\varepsilon}W[j]' \ \leq -i \end{aligned}$$

3 Convergence

Convergence of the algorithms depends entirely on stopping $W[j]$ from overflowing its register. It must be bounded. Define $Z \equiv 0.111...$ to have all its digits equal to 1. By taking representations $M \equiv M[j]+2^{-j}mZ$ for each input and $M \equiv M[j-1-\delta]+2^{1+\delta-j}mZ$ for each output and obtaining an expression for

$W[j{+}k]$ in terms of $W[j]$, it is possible to observe the worst case behaviour as $k \to \infty$. In the case of multiplication,

$$W[j{+}k] \quad = \quad W[j] + (2^k - 1)(W[j]{+}yX[j]{+}xY[j]{+}2^{1+\delta}p{+}2^{-j}xyZ[k])$$

Take $x = s_Y$ and $y = s_X$ where s_M is the sign of M. Then choosing $p = -1$ gives a lower bound on the maximum value of $W[j{+}k]$. So $W[j{+}k]$ is only bounded above as $k \to \infty$ if $W[j]$ is bounded above by

$$U[j] = 2^{1+\delta}{-}|X[j]|{-}|Y[j]|{-}2^{-j}.$$

Similarly, for the other two algorithms, there are upper bounds

$$\begin{array}{l} U[j] = 2^{1+\delta}|Y[j]| - 1 - |Q[j{-}1{-}\delta]| - 2^{1+\delta-j}s_Y s_Q \\ U[j] = 2^{2+\delta}|S[j{-}1{-}\delta]| + 2^{2+2\delta-j} - 1 \end{array}$$

for $j > \delta$ in each case. Lower bounds are essentially identical with opposite signs. These bounds are very tight in practice, and so aid in the generation of divergent examples since, when they are exceeded, an appropriate choice of repeated digit for the inputs will result in the partial remainder becoming arbitrarily large. Also, an easy consequence is that the hardware register for W is bounded by $|W| < 2^{1+\delta}$ for multiplication and division, and by $|W| < 3.2^{1+\delta}$ for square root.

Though tricky at some points, it is now possible to show by induction on j that these bounds do hold, subject to certain restrictions which are sufficient to make the induction step work. The base case of $j = 1{+}\delta$ holds because the integral part of the output is set to 0. As an example of the induction step, take multiplication when $p_{j-1-\delta} = 0$. For convenience, assume also that $\varepsilon < \delta$. Then $W[j{-}1]$ has at most $j{-}1$ digits after the point. So $W[j{-}1]{+}2^{1-j} \le W[j{-}1]'{+}2^{-\varepsilon} \le 2^{-\varepsilon}i$, and

$$\begin{array}{rl} W[j] & = 2W[j{-}1] + y_jX[j{-}1] + x_jY[j{-}1] + 2^{-j}x_jy_j - 2^{1+\delta}p_{j-1-\delta} \\ & \le 2(2^{-\varepsilon}i{-}2^{1-j}) + 2 \\ & \le 2^{1-\varepsilon}i - 2^{2-j} + 4 - |X[j]| - |Y[j]| \end{array}$$

This is at most $U[j]$ whenever $2^{1-\varepsilon}i{+}4 \;\le\; 2^{1+\delta}{+}2^{1-j}$, and so the step holds for all $j > \delta$ if

$$2^{1-\varepsilon}i \;+\; 4 \quad \le \quad 2^{1+\delta}$$

Similarly, the case $p_{j-1-\delta} = -1$ holds if

$$2^{1-\varepsilon}(1{-}i) \;+\; 4 \quad \le \quad 0$$

For division and square root convergence is guaranteed when

$$\begin{array}{ll} 2^{1-\varepsilon}i + 4 & \le 2^{\delta} \\ 2^{-\varepsilon}(1{-}i) + 2 & \le 0 \end{array}$$

and

$$\begin{array}{ll} 2^{1-\varepsilon}i + 2 & \le 2^{1+\delta} \\ 2^{1-\varepsilon}(1{-}i) + 2 & \le 0 \end{array}$$

hold respectively. The best choices of parameters which satisfy these have (δ, ε) equal to $(3, -1)$, $(4, -1)$ and $(2, 0)$, respectively, with i equal to 2 or 3 in each case.

4 Stable Number Representations

Our main question now is, are these conditions *necessary* for convergence? It is clear by exhaustive testing of all possible inputs that if word length is limited some better choices for ε and δ *can* be found. (Just consider a word length of one digit after the point to see that δ could then be reduced to 1 with $\varepsilon = 1$.) We will not try to establish at what word length the values of δ and ε need to be incremented, although some upper limits can be deduced from the examples below. However, it will be shown from examples which diverge, that the values obtained in the previous section are indeed the best ones when word length is unbounded or unspecified.

Divergence depends on certain choices of output digit, and therefore on the representation of the partial remainder, which in turn depends on the choice of hardware adder used in the iterative cycle. Prediction of representations is easiest when we have a *stable* representation, i.e. one that is invariant as it goes through the adder with the other inputs being identically 0. If we consider common hardware for the algorithms, then we need an adder for 3 redundant numbers, which we now define. For convenience, assume all digit positions behave identically, although any hardware is allowable for the un-truncated part of interest in W. First, the addition of three corresponding digits gives a total t which generates a sum digit $(|t| mod 2)\times sign(t)$ and a carry digit $(|t| div 2)\times sign(t)$. Secondly, corresponding sum and carry digits are added to yield a total t'. This is split in the same way again unless both it and the next digit total down are non-zero with the same sign, in which case the new sum digit is $-t'$ and the new carry is t'. Lastly, these second sum and carry results are added to give a final digit sum in the range $[-1..+1]$. This is the adder output.

Lemma 1.
- i) Every number has a stable representation.
- ii) A representation is stable if, and only if, it contains no pair of digits 11 or $\bar{1}\bar{1}$.
- iii) The largest stable fraction is $0.1010... = 2/3$.
- iv) The largest fraction none of whose stable representations has non-zero integral part is $0.01010... = 1/3$.
- v) If the binary representation of V has most significant digit corresponding to 2^n then stable representations have most significant digits corresponding to 2^n, 2^{n+1} or 2^{n+2}.
- vi) A finite representation of V whose binary representation has v digits becomes stable after at most $v+1$ passes through the adder, and cannot grow in length except from v to $v+1$ digits.

Lemma 2. For the truncation function $'$ and a stable representation V,
- i) If $V' > 0$ then $0 \leq (V/2)' \leq V'$;
- ii) If $V' = (V/2)' > 0$ then $V' \equiv 1\bar{1}\times 2^{-\varepsilon}$.

5 Values for the Partial Remainder

Assume each algorithm converges. The examples we now construct to obtain a contradiction to this assumption employ inputs which are are close to the points at which the induction argument above was most difficult to close, namely where the input digits have the greatest effect, and the output digit the least effect, on the value of the partial remainder. To obtain known output, we exploit the fact that if some choice of output were to yield $|W| < 1$ then any other choice of output would make W exceed its bounds. Hence the algorithm must make that first choice. Moreover, by Lemma 1(vi), inputs can be chosen to force stable representations of W.

Let $=$ denote equality of value between two possibly different representations and $\equiv$ equality of representation. As above, take $Z \equiv 0.111...$ and suppose V is the representation of some number to be chosen later, with most significant digit of index not above $-\delta$ and least significant digit of index $v \geq 0$.

For multiplication, take

$$X \;\equiv\; Z[t] + 2^{-u}V + 2^{-r-u}Z \quad \text{and} \quad Y \;\equiv\; Z[t] + 2^{-r} + 2^{-r-u}Z$$

where $u > t{+}1{+}2\delta$ and $r > u{+}v$. For these inputs, if the expected value for $P[r{+}u{-}2{-}2\delta]$ is chosen then $|W[r{+}u{-}1{-}\delta]| = |2^{-1-\delta}V| < 1$. Since any other choice for P would exceed the convergence bounds, this must indeed be the value for $P[r{+}u{-}2{-}2\delta]$. So, if V is initially stable and not large enough to generate earlier non-zero output digits, we obtain it as the value and representation of $W[r{+}u]$.

For division, take

$$X \;\equiv\; 2^{-1}Z[t] - 2^{-u-1}V + 2^{-r-u}Z \quad \text{and} \quad Y \;\equiv\; 2^{-1} + 2^{-r} - 2^{-r-u}Z$$

where $u > t{+}v{+}2\delta$ and $r > u{+}v$. A similar argument shows that $Q[r{+}t{-}1] = 1{-}2^{-t}{+}\ 2^{-u}V{-}2^{1-r}{+}2^{1-r-t}$, which yields $W[r{+}t{+}\delta] = 2^{t+\delta-u}V + 2^{t+1+\delta-r} + 2^{1+\delta-r}$. There are sufficient further iterations for the adder to turn V into a stable form when it appears in $W[r{+}u] = V{+}2^{u+1-r}{+}2^{u+1-r-t}$, subject to being sufficiently small not to generate earlier non-zero output digits.

For the square root, take

$$X \;\equiv\; 2^{-2} + 2^{-t} + 2^{-2t} - 2^{-u}V + 2^{1-u-t}Z$$

where $t > 2(v{+}1{+}\delta)$ and $u > 2t{+}2{+}2\delta$. As before, $S[u{+}v] = 2^{-1}{+}2^{-t}{-}2^{-u}V$ must hold, giving $W[u{+}t{-}1] = V{+}2^{t-u-1}V^2$ where the representation of V has become stable, and we assume it is sufficiently small not to generate earlier output digits.

In general, it is thus possible to obtain any number as the value of the partial remainder subject to certain conditions about one or all of its stable representations not generating earlier output digits.

6 Examples of Divergence

Two values for V are now chosen to make the value of W as close as possible to the points at which the output digit changes from 0 to +1 and from −1 to 0. These are where the induction step was hardest to establish. In particular, for i as in the definition of the output digit, we take

$$\begin{aligned} V_0 &= 2^{-\varepsilon}(i-1+\alpha) \\ V_{-1} &= 2^{-\varepsilon}(\beta-i) \end{aligned}$$

where α and β are fractions such that $V_0' = 2^{-\varepsilon}(i-1)$ and $V_{-1}' = -2^{-\varepsilon}i$ for the representations of V which might turn up in W. This will guarantee the output digits 0 and −1 respectively when desired, without $2^{-j}V$ $(j>0)$ generating non-zero output integers unless we obtain $V_{-1}' \equiv 1\bar{1}\times 2^{-\varepsilon}$ when $i=1$. With that exception, running through the induction step argument again, we find that the bound is *not* met if either of the following holds:

$$\begin{aligned} 2^{1-\varepsilon}(i-1+\alpha) + 4 &> 2^{\delta} \\ 2^{1-\varepsilon}(\beta-i) \quad + 4 &> 0 \end{aligned}$$

or

$$\begin{aligned} 2^{1-\varepsilon}(i-1+\alpha) + 4 &> 2^{\delta-1} \\ 2^{1-\varepsilon}(\beta-i) \quad + 4 &> 0 \end{aligned}$$

or

$$\begin{aligned} 2^{1-\varepsilon}(i-1+\alpha) + 2 &> 2^{\delta} \\ 2^{1-\varepsilon}(\beta-i) \quad + 2 &> 0 \end{aligned}$$

respectively for the three algorithms.

The best choice of α and β here would be to take them both equal to 1. However, some separation is needed between them and the “integer” part of V in order to ensure that the truncation function generates the right output digit. For multiplication, the initial representation of V is stable (avoiding the exceptional case) and, according to Lemma 1, α and β can be 1/3 or 2/3 (or rather the first v digits of their stable binary representations) depending on whether the integer part is odd or even respectively. Immediately, one of the inequalities is satisfied when $\delta = 2$, so that the multiplication algorithm must diverge. For $\delta = 3$, apart from the convergent cases we already know, the three cases $(\varepsilon, i) = (-3,1)$, $(-2,1)$, or $(-2,2)$ are not shown to be divergent by the above, but then divergence happens for

$$X \;\equiv\; Y \;\equiv\; Z[t] + 2^{-t-1}Z \;=\; 1 - 2^{-t-1}$$

with $t > 2$.

For division, we do not know which stable representation of V will appear in W. To guarantee the right output digits, a zero must be placed between any stable representations of the integer and fractional parts to prevent any interchange between them. So we may only choose α and β to be 1/6 or 1/3 depending on whether the integer part is odd or even respectively. Then with several exceptions, one of the inequalities is satisfied whenever $\delta \leq 4$, so that the division algorithm must diverge. Apart from the known convergent cases, the remaining undetermined exceptions are $(\varepsilon, i) = (-2, 1)$ for $\delta = 3$, and $(\varepsilon, i) = (-4, 1)$, $(-3, 1)$, $(-2, 1)$, or $(-2, 2)$ for $\delta = 4$. For these, modify the standard construction to take

$$X \;\equiv\; 2^{-1}Z[t] - 2^{-u-1}V + 2^{1-r-u}Z$$

with Y as before. The best β now works for the $\delta = 3$ case, whereas in the other cases, taking $V = 1{+}2^{-a}$ for some $a > 2$ produces an excessive W after an extra two or three iterations (or earlier in the case of the first exception). If the exception noted in Lemma 2 arises, then Y should be modified to

$$Y \;\equiv\; 2^{-1} + 2^{-r-1} - 2^{-r-u}Z$$

With the square root, the same choice of α and β is made as similar circumstances arise to those of division. With a few exceptions, one of the inequalities is then satisfied whenever $\delta \leq 2$, so that the division algorithm must diverge. Other than the known convergent cases, the exceptions are $(\varepsilon, i) = (-1, 1)$ for $\delta = 1$, and $(\varepsilon, i) = (-2, 1)$, $(-1, 1)$ or $(-1, 2)$ for $\delta = 3$. For both the cases $(-1, 1)$ the usual inequality will hold by choosing $V \equiv \bar{1}{+}2^{-a}$, with a not too small, unless the Lemma 2 exception arises, in which case $V \equiv \bar{1}0{+}2^{-a}$ will work. For the last case, $(-1, 2)$, consider $W[u{+}3]$ when

$$X \;\equiv\; 2^{-2} + 2^{-t} + 2^{-2t} - 2^{-u} + 2^{-1-u}Z$$

for t and $u > 2t$ large enough, say $t = 6$ and $u = 27$. The most interesting case is the second one, $(-2, 1)$, where all 15 digit inputs except one are calculated properly. The unique exception is $X \equiv 0.01010\bar{1}0\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}{+}2^{-15}Z$ for which the first incorrect digit has index 22. Divergence becomes guaranteed since $W[25] > U[25]$. Thus, this set of parameters is acceptable for all inputs up to at least 15 and perhaps up to as many as 21 digits.

7 Conclusion

We have investigated the optimality of the best parameters generated by analytic arguments for some on-line algorithms, and given techniques and general constructions for examples of divergence for other parameter choices. This shows the analytic results are indeed optimal. However, for the square root algorithm this is only just the case. Indeed, for limited precision square root better parameters can be chosen. Moreover, there seems no theoretical barrier to the existence of on-line functions for which the analytic results provide only strictly *sub*optimal parameters.

Two final remarks: shifting of difficult inputs can allow improved parameters, and, although the proofs become more complex, all the results hold when the inputs lie in the closure of the (half-open) intervals on which they are usually defined.

References

[1] Jean Duprat, Yvan Herreros & Jean-Michel Muller, "Some results about on-line computation of functions", *Proc. 9th Symposium on Computer Arithmetic*, Santa Monica CA, 6-8 Sept '89, IEEE Comp. Soc. Press, 1989, pp. 112-118.

[2] Milos D. Ercegovac, "An On-Line Square Rooting Algorithm", *Proc. 4th IEEE Symposium on Computer Arithmetic*, Santa Monica, CA, 1978, pp. 183-189.

[3] Milos D. Ercegovac & Tomas Lang, "Implementation of Module combining Multiplication, Division and Square Root", *Proc. IEEE Intern. Symp. on Circuits and Systems*, 1989, pp. 150-153.

[4] Milos D. Ercegovac & Tomas Lang, "On-Line Arithmetic: A Design Methodology and Applications in Digital Signal Processing", *VLSI Signal Processing III*, R. W. Brodersen, H. S. Moscovitz eds., IEEE Press, New York, 1988, pp. 252-263.

[5] V. G. Oklobdzija & Milos D. Ercegovac, "An On-Line Square Root Algorithm", *IEEE Trans. Comput.*, vol. **C-31**, Jan. 1982, pp. 70-75.

[6] H. J. Sips & H. X. Lin, " A New Model for On-Line Arithmetic with an Application to the Reciprocal Calculation", *J. Parallel & Distrib. Comp.* vol. **8**, 1990, pp. 218-230.

[7] Kishor S. Trivedi & Milos D. Ercegovac, "On-Line Algorithms for Division and Multiplication", *IEEE Trans. Comput.*, vol. **C-26**, No 7, July 1977, pp. 681-687.

[8] Paul K.-G. Tu & Milos D. Ercegovac, "Design of On-Line Division Unit", *Proc. 9th Symposium on Computer Arithmetic*, Santa Monica CA, 6-8 Sept '89, IEEE Comp. Soc. Press, 1989.

[9] J. H. Zurawski & J. B. Gosling, "Design of a high-speed square root, multiply and divide unit", *IEEE Trans. Comput.*, vol. **C-36**, 1987, pp. 13-23.