On-Line Algorithms for Computing Exponentials and Logarithms

Asger Munk Nielsen[1] and Jean-Michel Muller[2]

[1] Dept. of Mathematics and Computer Science, Odense University, Denmark
[2] Laboratoire LIP and CNRS,Ecole Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France

Abstract. We propose a new on-line algorithm for fast evaluation of logarithms and exponentials. This algorithm is derived from the widely studied Briggs-De Lugish iteration. We examine various compromises between the on-line delay and the size of the required comparison constants.

1 Introduction

The point at stake in this paper is the design of algorithms that rapidly evaluate exponentials and logarithms. We focus on algorithms that belong to the *shift-and-add* class. Those algorithm use very simple elementary steps: additions, and shifts (i.e. multiplications by a power of the radix of the number system being used), and they go back to the 17th century: Briggs, a contemporary of Neper, invented an algorithm that made it possible to build the first tables of logarithms. For instance, to compute $\ln x$ in radix-2 arithmetic, numerous methods (including that of Briggs, adapted to this radix) [9, 5, 10, 12] consist of finding a sequence $d_k = -1, 0, 1$, such that $x \prod_{k=1}^{n}(1 + d_k 2^{-k}) \approx 1$. Then $\ln(x) \approx -\sum_{k=1}^{n} \ln(1 + d_k 2^{-k})$. Another method belonging to the shift-and-add class is the CORDIC algorithm, introduced in 1959 by J. Volder [14] and then generalized by J. Walther [15]. Such algorithms have been implemented in many pocket calculators or math co-processors (such as the Intel 8087 and the following Intel chips until the 486 family).
In his Ph.D. dissertation [13], N. Takagi suggests an algorithm for computing logarithms and exponentials that is adapted to the use of redundant number systems (redundant number systems [2, 8] allow very fast carry-free additions, but they may complicate the comparisons that are required to perform the iterations: this point will be made clearer in what follows). We propose here a variant of that algorithm that allow on-line computation of logarithms and exponentials. On-line arithmetic was introduced by Ercegovac and Trivedi [7]. In on-line arithmetic, the digits flow through the operators in a digit-serial fashion, most significant digit first. This makes it possible to get a high throughput by pipelining the arithmetic operators at a digit-level. On line algorithms have been proposed for the arithmetic operations [7, 4, 3] as well as for computing elementary functions [11, 6]

Our algorithm is based on the following iteration :

$$\begin{cases} E_{n+1} = E_n - ln(1 + d_n 2^{-n}) \\ L_{n+1} = L_n(1 + d_n 2^{-n}) \end{cases} \quad (1)$$

if we take $E_1 = X$ and $L_1 = Y$ then (1) satisfies, $E_{n+1} = X - \sum_{k=1}^{n} ln(1+d_k 2^{-k})$ and $L_{n+1} = Y \prod_{k=1}^{n}(1 + d_k 2^{-k})$.

- If we are able to find a sequence of digits $d_k \in \{-1, 0, 1\}$ such that E_n goes to zero, then we will have $L_n \to Y e^X$. Thus we have computed the product of the number Y and the *exponential* of the number X. This iteration mode will be referred to as *mode zero.*
- If we are able to find a sequence of digits $d_k \in \{-1, 0, 1\}$ such that L_n goes to one, then we will have $E_n \to X + ln(Y)$. Thus we have computed the *logarithm* of the number Y added to the number X. This iteration mode will be referred to as *mode one.*

This paper is an abridged version of a longer technical report. The proofs of the claims stated in the following sections can be found along with further details in [1].

2 Selection of Digits and Domain of Convergence

2.1 Computing Exponentials

In the *zero mode* the digits d_k must be chosen such that the iteration variable E_n goes to zero. By plotting E_{n+1} versus E_n for all values of $d_n \in \{-1, 0, 1\}$, as according to $E_{n+1} = E_n - ln(1 + d_n 2^{-n})$, we get a diagram as depicted in Fig. 1. This diagram is similar to the *Robertson Diagrams* frequently found in studies of division algorithms. If we assume that $E_1 = X$ belongs to some interval $[l_1, u_1]$

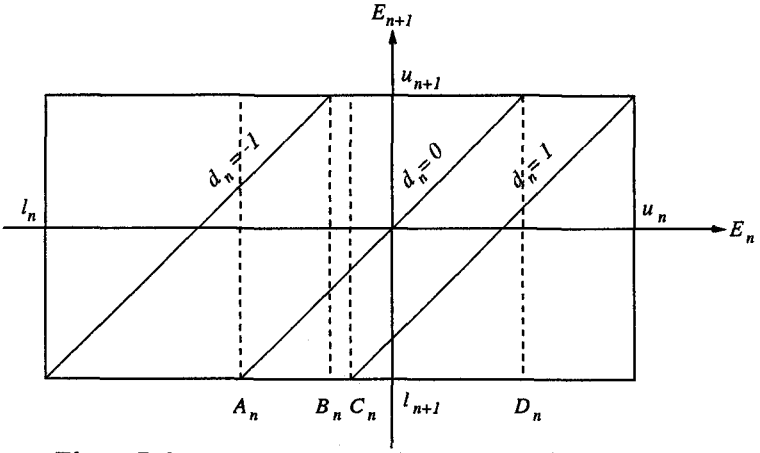

Fig. 1. Robertson diagram for the zero mode (exponentials).

and choose the digits d_k such that E_{k+1} belongs to some interval $[l_{k+1}, u_{k+1}]$, where the sequence of intervals is such that:

$$[l_1, u_1] \supset [l_2, u_2] \supset \cdots \supset [l_n, u_n], \tag{2}$$

where each of these intervals contain zero and $\lim_{n\to\infty} u_n - l_n = 0$, then E_n will exhibit the desired convergence towards zero. From Fig. 1 we may deduce a proper convergence domain and digit selection procedure for the *zero mode*. By noting that $u_{n+1} = u_n - ln(1+2^{-n})$ and $l_{n+1} = l_n - ln(1-2^{-n})$, and since u_n and l_n must go to zero as n goes to infinity, we deduce that $u_n = \sum_{k=n}^{\infty} ln(1+2^{-k})$ and $l_n = \sum_{k=n}^{\infty} ln(1-2^{-k})$. From Fig. 1 we may further deduce a valid selection procedure for the digits d_k as:

$$d_k = \begin{cases} -1 & \text{if } E_n \leq B_n \\ 0 & \text{if } A_n \leq E_n \leq D_n \\ 1 & \text{if } C_n \leq E_n \end{cases} \tag{3}$$

Notice that the selection procedure is *nondeterministic* in the sense that for some values of E_n several choices of d_n may be valid. The constants A_n, B_n, C_n and D_n are equal to (again these values are deduceable from the Robertson Diagram): $A_n = l_{n+1}$, $B_n = u_{n+1} + ln(1-2^{-n})$, $C_n = l_{n+1} + ln(1+2^{-n})$, $D_n = u_{n+1}$. It can be shown that $A_n < B_n < C_n < D_n$ for all $n \geq 1$. Refer to Tab. 1 for the four first values and limit of these constants. The convergence domain of the algorithm is given by the requirement that $E_1 = X \in [l_1, u_1]$, from which we deduce that $-1.24206... = l_1 \leq X \leq u_1 = 0.86887....$

n	$2^n l_n$	$2^n u_n$	$2^n A_n$	$2^n B_n$	$2^n C_n$	$2^n D_n$
1	-2.484123	1.737752	-1.097829	-0.4594709	-0.2868995	0.9268231
2	-2.195658	1.853648	-1.044930	-0.1896560	-0.1523566	0.9610720
3	-2.089862	1.922144	-1.021611	-0.0883713	-0.0793469	0.9798797
4	-2.043223	1.959760	-1.010607	-0.0428505	-0.0406114	0.9897655
∞	-2	2	-1	0	0	1

Table 1. First 4 values and limits of $2^n l_n, 2^n u_n, 2^n A_n, 2^n B_n, 2^n C_n$ and $2^n D_n$.

The relative error is bounded by:

$$-2\cdot 2^{-n} < \prod_{k=n}^{\infty}(1-2^{-k}) - 1 = e^{l_n} - 1 \leq \frac{Ye^X - L_n}{L_n} \leq e^{u_n} - 1 = \prod_{k=n}^{\infty}(1+2^{-k}) - 1 < 2.8\cdot 2^{-n} \tag{4}$$

2.2 Computing Logarithms

In the *one mode* the digits d_k must be chosen such that the iteration variable L_n goes to one. By plotting $\lambda_{n+1} = L_{n+1} - 1$ versus $\lambda_n = L_n - 1$ for all values

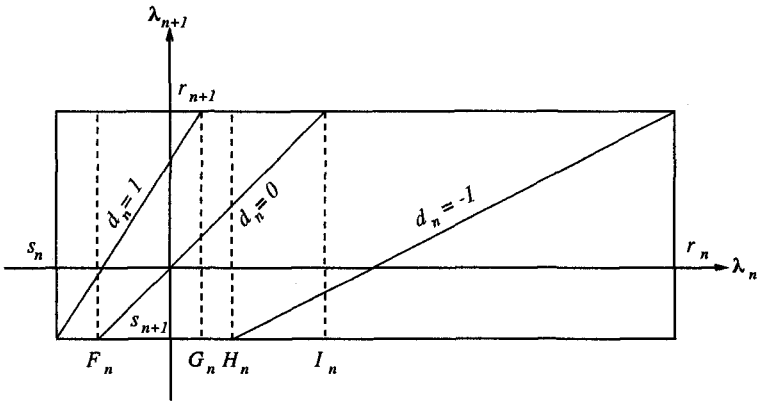

Fig. 2. Robertson diagram for the one mode (logarithms, $n = 1$).

of $d_n \in \{-1, 0, 1\}$, according to $\lambda_{n+1} = L_{n+1} - 1 = \lambda_n(1 + d_n 2^{-n}) + d_n 2^{-n}$, we get a diagram as depicted in Fig. 2. If we assume that $\lambda_1 = L_1 - 1 = Y - 1$ belongs to some interval $[s_1, r_1]$, and that the digits d_k are chosen such that $\lambda_{k+1} = L_{k+1} - 1$ belongs to some interval $[s_{k+1}, r_{k+1}]$, where these intervals include zero and are contracting in the sense that the width of the interval goes to zero, then L_n will converge towards one, as n approaches infinity. From Fig. 2 we may deduce a proper convergence domain and selection procedure for the *one mode*. By noting that $r_{n+1} = (1 - 2^{-n})r_n - 2^{-n}$ and $s_{n+1} = (1 + 2^{-n})s_n + 2^{-n}$, and since r_n and s_n must go to zero as n goes to infinity, we deduce that $r_n = \sum_{k=n}^{\infty} 2^{-k} \prod_{j=n}^{k} (1 - 2^{-j})^{-1}$ and $s_n = -\sum_{k=n}^{\infty} 2^{-k} \prod_{j=n}^{k} (1 + 2^{-j})^{-1}$. From Fig. 2 we may further deduce a selection procedure for the digits d_k as:

$$d_k = \begin{cases} 1 & \text{if } L_n - 1 \leq G_n \\ 0 & \text{if } F_n \leq L_n - 1 \leq I_n \\ -1 & \text{if } H_n \leq L_n - 1 \end{cases} \tag{5}$$

Again this selection procedure is nondeterministic. The constants F_n, G_n, H_n and I_n are equal to:

$$F_n = s_{n+1}\ ,\ G_n = \frac{r_{n+1} - 2^{-n}}{1 + 2^{-n}}\ ,\ H_n = \frac{s_{n+1} + 2^{-n}}{1 - 2^{-n}}\ ,\ I_n = r_{n+1} \tag{6}$$

It can be shown that $F_n < G_n < H_n < I_n$ for all $n \geq 1$. Refer to Tab. 2 for the first four values and limit of these constants. The convergence domain of the algorithm is given by the requirement that $\lambda_1 = L_1 - 1 = Y - 1 \in [s_1, r_1]$, from which we deduce that $0.4194... = s_1 + 1 \leq Y \leq u_1 + 1 = 3.4627....$

The absolute error is given by:

$$-2 \cdot 2^{-n} \leq ln(s_n + 1) \leq (X + ln(Y)) - E_n \leq ln(r_n + 1) \leq r_1 2^{-n} < 2.5 \cdot 2^{-n} \tag{7}$$

3 On-Line Algorithms

By definition, an on-line algorithm should be able to deduce the nth digit of the result, based on $\delta + n$ digits of the operands, where the integer δ is referred to as

n	$2^n s_n$	$2^n r_n$	$2^n F_n$	$2^n G_n$	$2^n H_n$	$2^n I_n$
1	-1.161155	4.925493	-0.7417327	0.3084977	0.5165347	1.462747
2	-1.483465	2.925493	-0.8543317	0.1552959	0.1942244	1.194120
3	-1.708663	2.388240	-0.9222463	0.0797421	0.0888614	1.089710
4	-1.844493	2.179420	-0.9597734	0.0406645	0.0429084	1.043206
∞	-2	2	-1	0	0	1

Table 2. First 4 values and limits of $2^n s_n, 2^n r_n, 2^n F_n, 2^n G_n, 2^n H_n$ and $2^n I_n$.

the *on-line delay* [7]. Thus in order to compute the exponential and logarithm of a number, in an on-line manner, we must develop algorithms that compute the desired value based on limited knowledge of the input operands X and Y.

We will devise such an algorithm, and prove that it emulates a run of the nondeterministic algorithms described in the previous section. The algorithm is based on two scaled iteration variables $\hat{E}_n = 2^n(X_{n+\delta} - \sum_{k=1}^{n-1} ln(1 + \hat{d}_k 2^{-k}))$ and $\hat{L}_n = 2^n(Y_{n+\delta} \prod_{k=1}^{n-1}(1 + \hat{d}_k 2^{-k}) - m)$, where m is set to either zero or one, corresponding to the desired mode of computation. Based on these iteration variables the proper digits $\hat{d}_k$ can be selected, as specified by the following algorithm.

Algorithm 1. *On-line Exponentials and Logarithms.*
Stimulus: m : *Boolean (zero or one mode),* $X \in [l_1, u_1] = [-1.24206..., 0.86887...]$ *(if* $m = 0$*),* $Y \in [s_1 + 1, r_1 + 1] = [04194..., 3.4627...]$ *(if* $m = 1$*).*
Response: If $m = 0$ *then* $\lim_{n\to\infty} 2^{-n}\hat{L}_n = Ye^X$. *If* $m = 1$ *then* $\lim_{n\to\infty} 2^{-n}\hat{E}_n = X + ln(Y)$.

1. *Initialize:* $\hat{E}_0 = x_0.x_1 \cdots x_\delta$, $\hat{L}_0 = y_{-1}y_0.y_1 \cdots y_\delta - m$, $P_0 = 1$ *and* $\hat{d}_0 = 0$.
2. *iterate: (for* $n = 1, 2, ...$*)*

$$\hat{E}_n = 2\hat{E}_{n-1} + x_{n+\delta}2^{-\delta} - 2^n ln(1 + \hat{d}_{n-1}2^{-n+1})$$
$$P_n = P_{n-1}(1 + \hat{d}_{n-1}2^{-n+1})$$
$$\hat{L}_n = (2\hat{L}_{n-1} + y_{n+\delta}P_{n-1}2^{-\delta})(1 + \hat{d}_{n-1}2^{-n+1}) + 2md_{n-1}$$

If $m = 0$ *then compute* $\tilde{E}_n = trunc(\hat{E}_n, p)$ *and* $\hat{d}_k = \alpha$, *else* $\tilde{L}_n = trunc(\hat{L}_n, q)$ *and* $\hat{d}_k = \gamma$ *where*

$$\alpha = \begin{cases} -1 & if\ \tilde{E}_n \le A \\ 0 & if\ A \le \tilde{E}_n \le B \\ 1 & if\ B \le \tilde{E}_n \end{cases}, \gamma = \begin{cases} 1 & if\ \tilde{L}_n \le C \\ 0 & if\ C \le \tilde{L}_n \le D \\ -1 & if\ D \le \tilde{L}_n \end{cases}$$

The function $trunc(x, p)$ referred to in the algorithm, truncates the value x to p fractional digits. Rather than actually computing the term $2^n ln(1 + \hat{d}_{n-1}2^{-n+1})$, this term should be accessible by means of a table lookup. The following lemma gives bounds on the comparison constants A, B, C and D, for which Alg. 1 selects the digits $\hat{d}_k$ correctly.

Lemma 2. Correctness of digit selection.
If A, B and C, D are p respectively q fractional digit constants satisfying for all $n \geq 1$:

$$2^n A_n + 2^{-p} + 2^{-\delta} \leq A \leq 2^n B_n - 2^{-p} - 2^{-\delta} \tag{8}$$

and

$$2^n C_n + 2^{-p} + 2^{-\delta} \leq B \leq 2^n D_n - 2^{-p} - 2^{-\delta} \tag{9}$$

respectively

$$2^n F_n + 2^{-q} + 2^{-\delta} \prod_{k=1}^{n-1}(1+2^{-k}) \leq C \leq 2^n G_n - 2^{-q} - 2^{-\delta} \prod_{k=1}^{n-1}(1+2^{-k}) \tag{10}$$

and

$$2^n H_n + 2^{-q} + 2^{-\delta} \prod_{k=1}^{n-1}(1+2^{-k}) \leq D \leq 2^n I_n - 2^{-q} - 2^{-\delta} \prod_{k=1}^{n-1}(1+2^{-k}) \tag{11}$$

then given the input X and Y, Alg. 1 will emulate a run of the nondeterministic parallel algorithm (in the sense that the nondeterministic algorithm may choose the same digits as the ones chosen by the on-line algorithm, i.e. $d_k = \hat{d}_k$), when supplied with the same input.

Corollary 3. *Under assumption of the hypothesis of Lemma 2, we have:*

$$\left| L_n - 2^{-n}\hat{L}_n \right| < \left(\prod_{k=1}^{n-1}(1+2^{-k})\right)2^{-n-\delta} \text{ and } \left| E_n - 2^{-n}\hat{E}_n \right| < 2^{-n-\delta}. \tag{12}$$

for mode zero respectively one.

3.1 Hardware Requirements

The algorithm requires the storage of the constants $2^j ln(1 + \hat{d}_{j-1}2^{-j+1})$ for $\hat{d}_{j-1} \in \{-1, 0, 1\}$, thus to obtain an accuracy of the result of approximately n binary digits, we need to store $3n$ constants. This result can be improved by using the fact that for $j \geq n/2$, $ln(1 + d_j 2^{-j})$ can be replaced by $d_j 2^{-j}$ with accuracy 2^{-n}. Thus we only need to store $\frac{3}{2}n$ constants.

The updating of the iteration variables, as specified by (8), can be done efficiently using only shifts and additions. If the step time of each iteration of the algorithm is to be independent of the desired accuracy, e.g. the number of binary digits, the additions must be performed using *redundant arithmetic*, and the variables $\hat{E}_n$, P_n and $\hat{L}_n$ should be represented in a redundant representation, like borrow-save. One inherent drawback of redundant arithmetic is that when comparing two redundant numbers, we are forced to examine all digits of both numbers. By designing the digit selection of Alg. 1, such that the comparisons are based on truncated redundant numbers, this drawback has been eliminated, in the sense that we only need to examine a constant number of digits of both operands.

4 Choice of Constants and On-line Delay

In this section we will discuss how to choose the comparison constants A, B, C and D, used when selecting the digits $\hat{d}_k$ in the on-line algorithm. The lower and upper bounds for the constants A and B, as given by Lemma 2, are depicted in Fig. 3.

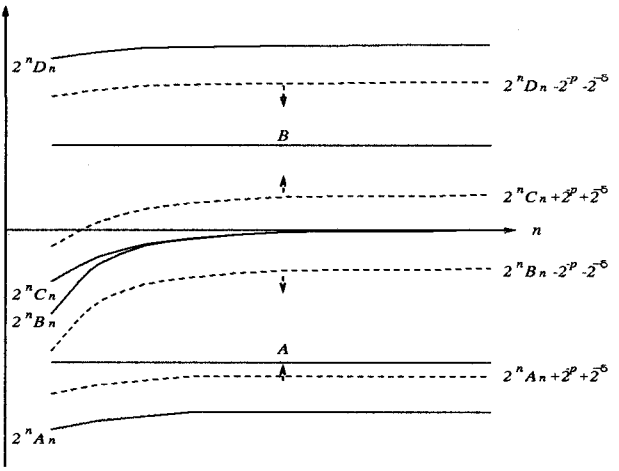

Fig. 3. Lower and upper bound for constants A and B.

By inspection of Fig. 3 we note that:

$$2^n A_n + 2^{-p} + 2^{-\delta} \leq \lim_{n\to\infty} 2^n A_n + 2^{-p} + 2^{-\delta} \leq A \leq 2^1 B_1 - 2^{-p} - 2^{-\delta} \leq 2^n B_n - 2^{-p} - 2^{-\delta} \quad (13)$$

thus the p fractional digit constant A, must be chosen such that it belongs to the interval

$$[\lim_{n\to\infty} 2^n A_n + 2^{-p} + 2^{-\delta}, 2^1 B_1 - 2^{-p} - 2^{-\delta}].$$

The best choice of A, will be the *midpoint* of this interval truncated to p fractional digits, e.g. $A = trunc(\frac{\lim_{n\to\infty} 2^n A_n + 2^{-p} + 2^{-\delta} + 2B_1 - 2^{-p} - 2^{-\delta}}{2}, p) = trunc(B_1 - \frac{1}{2}, p)$. In this way we have:

$$\left| A - (B_1 - \frac{1}{2}) \right| < 2^{-p}. \quad (14)$$

From (13) and (14) we get the requirement: $A < 2^{-p} + B_1 - \frac{1}{2} \leq 2B_1 - 2^{-p} - 2^{-\delta}$, from which we deduce:

$$\delta \geq \left\lceil \log_2 \left(\frac{1}{B_1 + 0.5 - 2^{-p+1}} \right) \right\rceil = \delta_A. \quad (15)$$

Similarly we choose the constant B as the midpoint of the interval $[\lim_{n\to\infty} 2^n C_n + 2^{-p} + 2^{-\delta}, 2^1 D_1 - 2^{-p} - 2^{-\delta}]$, e.g. $B = trunc(\frac{2D_1 + \lim_{n\to\infty} 2^n C_n}{2}, p) = trunc(D_1, p)$, and deduce that:

$$\delta \geq \left\lceil \log_2 \left(\frac{1}{D_1 - 2^{-p+1}} \right) \right\rceil = \delta_B. \quad (16)$$

Since $\delta_A \geq \delta_B$ the lower bound on-line delay when computing in mode zero is given by (15), e.g. $\delta \geq \delta_A$.

For *mode one*, a similar analysis can be performed, thereby determining values for the constants C, D and a lower bound on the on-line delay δ_D (see [1] for details). Examples of values of A, B, C, D and the on-line delays δ_A and δ_D, are given in Tab. 3 for various values of p and q.

p,q	δ_A	A	B	δ_D	C	D
3	6	-0.101	0.011	-	-	-
4	3	-0.1011	0.0111	4	-0.0110	0.1011
5	3	-0.10111	0.01110	4	-0.01101	0.10110
6	3	-0.101110	0.011101	3	-0.011101	0.101010
7	2	-0.1011101	0.0111011	3	-0.0111010	0.1010101
8	2	-0.10111010	0.01110110	3	-0.01110101	0.10101011
∞	2	-0.1011101011...	0.0111011010...	3	-0.0111010100...	0.1010101111...

Table 3. Truncated comparison comparison constants (in binary), and lower bound on on-line delay, for various values of p and q.

5 Digitization

The output of Alg. 1 is a sequence of approximations $\hat{L}_n$ or $\hat{E}_n$, to the exact result Ye^X respectively $X + ln(Y)$. If our algorithm for computing elementary functions is to be an on-line algorithm, we must produce the output as well as consume the input in an on-line manner. In order to produce the output in an on-line fashion, we will use a technique known as *digitization*.

Assume that we have computed a sequence of approximations $< \hat{Z}_n >$ to a number Z, such that $|2^{-n}\hat{Z}_n - Z| \leq 2^{-n+p}$, and that the numerical value of the exact result is bounded by $|Z| \leq 2^q$. We will then perform an on-line *digitization* of the value Z based on the sequence of approximations $< \hat{Z}_n >$ producing the result S_n in on-line mode. In each iteration of the algorithm, the (output) digit s_n is chosen such that the *residual error* $|\hat{Z}_n - S_n| = |\hat{Z}_n - 2S_{n-1} + s_n 2^c|$ is minimized.

Algorithm 4. *On-line Digitization.*
Stimulus: A sequence of approximations $<\hat{Z}_n>$ *satisfying:* $\left|2^{-n}\hat{Z}_n - Z\right| \leq 2^{-n+p}$.
Response: $S_n = 2^{c+1+n} \sum_{j=1}^{n} s_j 2^{-j}$.

1. *Initialize:* $S_0 = 0$

2. *iterate: (for* $n = 1, 2, ...$*)*
$T_n = trunc((\hat{Z}_n - 2S_{n-1})2^{-c}, k)$, $S_n = 2S_{n-1} + s_n 2^c$

$$s_n = \begin{cases} -1 & if\ T_n \leq -\frac{1}{2} \\ 0 & if\ -\frac{1}{2} < T_n \leq \frac{1}{2} \\ 1 & if\ \frac{1}{2} < T_n \end{cases}$$

The following lemma gives a bound on the residual error.

Lemma 5. *If the scaling constant* c *is chosen such that*

$$c \geq \max\left\{p + \left\lceil \log_2 \frac{1 + 2^{q-p+1}}{\frac{3}{2} + 2^{-k}} \right\rceil, p + \left\lceil \log_2 \frac{3}{\frac{1}{2} - 2^{-k}} \right\rceil\right\} \tag{17}$$

then the value S_j *computed by Alg.4 satisfies:* $\forall j \geq 1 : \left|\hat{Z}_j - S_j\right| \leq 2^c(\frac{1}{2} + 2^{-k})$.

From Lemma 5 we may deduce the following bound on the absolute error, made by approximating Z by $2^{-n}S_n$.

$$\begin{aligned} \left|Z - 2^{-n}S_n\right| &= \left|Z - 2^{-n}\hat{Z}_n + 2^{-n}\hat{Z}_n - 2^{-n}S_n\right| \leq \left|Z - 2^{-n}\hat{Z}_n\right| + 2^{-n}\left|\hat{Z}_n - S_n\right| \\ &\leq 2^{-n+p} + 2^c(\frac{1}{2} + 2^{-k})2^{-n} < 2^{c+1-n} \end{aligned}$$

Since the least significant digit of $2^{-n}S_n$ is of weight 2^{c+1-n}, we conclude that $2^{-n}S_n$ is within one *ulp* (unit last place) of the exact result Z.

Since the selection procedure is based on a k fractional digit estimation of $\hat{Z}_n - 2S_{n-1}$, it is not necessary to perform the full precision subtraction. Furthermore we only need to store S_n to k fractional digits. Consequently Alg. 4 can be implemented efficiently, with a hardware cost that is independent on the desired precision (the number of binary digits of the result).

5.1 Applying Digitization to the Output of On-line Exp/Log Algorithm.

In what follows we will briefly argue that Alg. 4 can be used to digitize the output of Alg. 1. For mode zero, by assuming $|Y| \leq 1$, we deduce from (7) and Corollary 3 that: $\left|2^{-n}L_n - Ye^X\right| < 2^{-n+3} = 2^{-n+p}$. The absolute value of the exact result can be estimated to: $\left|Ye^X\right| \leq |Y|e^{u_1} < 2^2 = 2^q$. Thus with $p = 3$ and $q = 2$, we get from Lemma 5 that the scaling constant c must satisfy, $c \geq 7$ for $k = 2$ and $c \geq 6$ for $k > 2$.

For mode one, we get from (4) and Corollary 3 that $\left|2^{-n}\hat{E}_n - (X + ln(Y))\right| < 2^{-n+2} = 2^{-n+p}$, and by assuming that $|X| \leq 1$ we get $|X + ln(Y)| \leq |X| + |ln(u_1 + 1)| < 2^2 = 2^q$. Thus with $p = 2$ and $q = 2$, we get from Lemma 5 that the scaling constant c must satisfy, $c \geq 6$ for $k = 2$ and $c \geq 5$ for $k > 2$.

6 Conclusion

We have proposed a new on-line algorithm for evaluating logarithms and exponentials. This algorithm is suitable for VLSI implementation. One should notice that in our presentation we rarely use the fact that the exponentials and logarithms are base e exponentials and logarithms, thus if one replaces the constants $ln(1+d_k 2^{-k})$ by $\log_a(1+d_k 2^{-k})$ then (by modifying the comparison constants), one will obtain an algorithm for computing $X + log_a Y$ and $Y a^X$.

References

1. Nielsen, A.M., Muller, J.M.: On-line Algorithms for Computing Exponentials and Logarithms. Technical report, available at "http://www.imada.ou.dk/Research/"
2. Avizienis, A.: Signed-digit number representations for fast parallel arithmetic. IRE Trans. on electronic comp., **10** (1961) 389–400
3. Bajard, J.C., Duprat, J., Kla, S., Muller, J.M.: Some operators for on-line radix 2 computations. Jour. of Parallel and Dist. Computing **22(2)** (1994) 336–345.
4. R.H. Brackert, R.H., Ercegovac, M.D., Willson, A.N.: Design of an on-line multiply-add module for recursive digital filters. In proc. of 9th Symp. on Computer Arithmetic, Santa Monica, USA, (1989) 34–41.
5. Chen, T.C.: Automatic computation of logarithms, exponentials, ratios and square roots. IBM J. of Res. and Dev. **16** (1972) 380–388.
6. Ercegovac, M.D., Lang, T.: On-line scheme for computing rotation factors. Jour. of Parallel and Dist. Comput., Special Issue on Parallelism in Computer Arithmetic (1988)
7. Ercegovac, M.D., Trivedi, K.S.: On-line algorithms for division and multiplication. IEEE Trans. on Comp. **C-26(7)** (1977) 681–687.
8. Koren, I.: Computer arithmetic algorithms Prentice-Hall (1993).
9. Lugish, B.De: A Class of Algorithms for Automatic Evaluation of Functions and Computations in a Digital Computer. PhD thesis, Dept. of Comp. Sci. Univ. of Illinois Urbana (1970).
10. Muller, J.M.: Discrete basis and computation of elementary functions. IEEE Trans. on Comp. **C-34(9)** (1985).
11. Oklobdzija, V.G., Ercegovac, M.D.: An on-line square root algorithm. IEEE Trans. on Comp. **C-31** (1982) 70–75.
12. Specker, W.H.: A class of algorithms for $ln(x)$, $exp(x)$, $sin(x)$, $cos(x)$, $tan^{-1}(x)$ and $cot^{-1}(x)$. IEEE Trans. on Elec. Comp. **EC-14** (1965)
13. Takagi, N.: Studies on hardware algorithms for arithmetic operations with a redundant binary representation. Ph.D thesis, Dept. Info. Sci., Kyoto Univ.1 (1987).
14. Volder, J. The cordic computing technique. IRE Trans. on Elect. Comp. (1959)
15. Walther, J.: A unified algorithm for elementary functions. In proc. of Joint Comp. Conf., (1971)