BKM: A New Hardware Algorithm for Complex Elementary Functions

Jean-Claude BAJARD, Sylvanus KLA and Jean-Michel MULLER

CNRS, Laboratoire LIP-IMAG, Ecole Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07 FRANCE

Abstract

A new algorithm for computing complex logarithms and exponentials is proposed. This algorithm is based on shift-and-add elementary steps, and it generalizes the CORDIC algorithm. It can compute the usual real elementary functions. This algorithm is more suitable for computations in a redundant number system than CORDIC, since there is no scaling factor for computation of trigonometric functions.

Index terms

Elementary functions, CORDIC.

1. Introduction

The point at stake here is the search for algorithms that rapidly compute elementary functions. Many methods have been used, e.g. approximation by polynomials, Newton's method, E-Method [4], and shift-and-add methods. The shift-and-add methods use simple elementary steps: additions, and multiplications by a power of the radix of the number system. They go back to the 17th century: Briggs used such an algorithm for building the first tables of logarithms [9]. For instance, in radix 2, to compute ln$x$ with approximately n significant bits, numerous methods [2], [7] consist of finding a sequence $d_k = \pm 1, 0$, 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 important shift-and-add method is the CORDIC algorithm, introduced by Volder [11] for computing trigonometric functions, and generalized by Walther [12]. CORDIC has been implemented in many machines (e.g. Hewlett Packard's HP 35, Intel 8087). It consists of the following iteration:

$$(1)\quad \begin{cases} x_{n+1} = x_n - m d_n y_n 2^{-\sigma(n)} \\ y_{n+1} = y_n + d_n x_n 2^{-\sigma(n)} \\ z_{n+1} = z_n - d_n e_{\sigma(n)} \end{cases}$$

m equals 0, 1 or -1, and d_n is equal to 1 or -1, so this iteration is reduced to a few additions and shifts. The results and the appropriate values of d_n, m and $\sigma(n)$ are given in Table 1. For a recent survey of CORDIC, see [5].

	rotation mode (d_n = sign z_n)	**vectoring mode** (d_n = - sign y_n)	$\sigma(n)$	e_n	**scale factor K**
m = 1 (circular)	$x_n \to K(x_0 \cos z_0 - y_0 \sin z_0)$ $y_n \to K(y_0 \cos z_0 + x_0 \sin z_0)$	$x_n \to K\sqrt{x_0^2 + y_0^2}$ $z_n \to z_0 + \tan^{-1} y_0/x_0$	n	$\tan^{-1}2^{-n}$	$\prod_{n=0}^{\infty}\sqrt{1+2^{-2n}} \approx 1.64676$
m = 0 (linear)	$x_n = x_0$ $y_n \to y_0 + x_0 z_0$	$x_n = x_0$ $z_n \to z_0 + y_0/x_0$	n	2^{-n}	no scale factor
m = -1 (hyperbolic)	$x_n \to K(x_0 \cosh z_0 + y_0 \sinh z_0)$ $y_n \to K(y_0 \cosh z_0 + x_0 \sinh z_0)$	$x_n \to K\sqrt{x_0^2 - y_0^2}$ $z_n \to z_0 + \tanh^{-1} y_0/x_0$	$\sigma(n) = n - k$, (k = largest integer such that $3^{k+1}+2k-1 \le 2n$)	$\tanh^{-1}2^{-n}$	$\prod_{n=1}^{\infty}\sqrt{1-2^{-2\sigma(n)}} \approx 0.82816$

Table 1. *Different functions computable using CORDIC.*

The major drawback of CORDIC arises when performing the iterations using a redundant number system. Such number systems are advantageous for quickly-performed arithmetic, since they make it possible to perform carry-free additions [1]. With these systems, d_n is difficult to evaluate. For instance, assume that we are in the *rotation* and *circular* modes of CORDIC (see Table 1), and that numbers are represented in radix 2 with digits in $\{-1, 0, 1\}$. d_n equals the sign of the most significant non zero digitof z_n: to find its value, we may have to examine all the digits of z_n, and the advantage of the redundant representation (constant time elementary step) is lost. An alternative is to accept $d_n = 0$,

but with such a method the scale factor K is no longer constant. $K = \prod_{n=0}^{\infty}\sqrt{1+d_n^2 2^{-2n}}$ is a constant if the d_i's are all equal to ± 1, but not if they can be 0. Many solutions have been suggested to solve this problem. They lead to a repetition of iterations in time [10], or in space [3]. To avoid this, we need to work out a new algorithm. Throughout this paper, we assume that we use a radix-2 usual or signed-digit number system. The main advantage of our algorithm (constant-time elementary step without scale factor) appears if the signed-digit system is used. Extension to binary carry-save representation is simple.

Consider the basic step of CORDIC in circular mode (i.e. (1) with m=1 and $\sigma(n)$=n), and define the complex number $L_n = x_n + iy_n$. We get: $L_{n+1} = L_n(1+id_n2^{-n})$. This brings us to a generalization of this algorithm: we could perform multiplications by terms of the form $(1+d_n2^{-n})$, where the d_n's are *complex numbers*, chosen such that a multiplication by d_n can be reduced to a few additions. In this paper, we study the following iteration, called *BKM*:

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

with d_n = -1, 0, 1, -i, i, 1-i, 1+i, -1-i, -1+i

ln z is the number $t = a+ib$ such that $e^t = e^a(\cos b + i \sin b) = z$, with b lying in $[-\pi,\pi)$.

If we find a sequence d_n such that L_n goes to 1, then we obtain $E_n \to E_1 + \ln(L_1)$: we call this iteration the ***L-mode*** of the BKM algorithm. If we find a sequence d_n such that E_n goes to 0, then we obtain $L_n \to L_1 e^{E_1}$: we call this iteration the ***E-mode*** of BKM. Therefore, in the next sections, we focus on the problem of finding sequences d_n such that L_n goes to 1 or such that E_n goes to 0.

2. Computation of the complex exponential function (E-mode)

For computing e^{E_1} using BKM, one needs to find a sequence $d_n \in D = \{-1,0,1,-i,i,i-1, i+1,-i-1,-i+1\}$ such that the sequence E_n of Eq. (2) goes to 0 as n goes to $+\infty$. At the outset, let us examine the numbers whose exponential can be computed. The set $A = \left\{\sum_{n=1}^{\infty}\ln\left(1+d_n2^{-n}\right) \mid d_n \in D\right\}$ of the numbers E_1 such that there exists a sequence $d_n \in D$ satisfying $E_n \to 0$ is shown in Fig. 1.

Define d_n^x and d_n^y as the real and imaginary parts of d_n and E_n^x and E_n^y as the real and imaginary parts of E_n. We find:

$$(3)\quad \begin{cases} E_{n+1}^x = E_n^x - \frac{1}{2}\ln\left[1+d_n^x 2^{-n+1} + \left(d_n^{x2} + d_n^{y2}\right)2^{-2n}\right] \\ E_{n+1}^y = E_n^y - d_n^y \tan^{-1}\left(\frac{2^{-n}}{1+d_n^x 2^{-n}}\right) \end{cases}$$

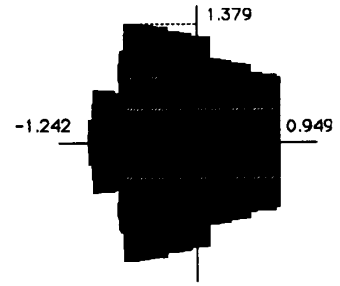

Fig.1. *The set A, and the convergence area of the E-mode.*

In this section, we give an algorithm which computes the sequence d_n for any E_1 belonging to a rectangular set R_1. The algorithm uses a sequence $R_n = [-s_n^x, r_n^x] + i[-r_n^y, r_n^y]$ of rectangles, whose length goes to 0 as n goes to $+\infty$, and such that for any E_n belonging to R_n, d_n is such that E_{n+1} belongs to R_{n+1}. d_n^x is chosen by examining a few digits of E_n^x and d_n^y is chosen by examining a few digits of E_n^y: this allows a simple implementation of the choice of d_n.

2.a. Choice of d_n^x

Fig 2 shows the parameters involved in determining d_n^x. This figure is close to the *Robertson Diagrams* that appear in many division algorithms [6], [8]. In the following, we call such a diagram a *Robertson diagram.*

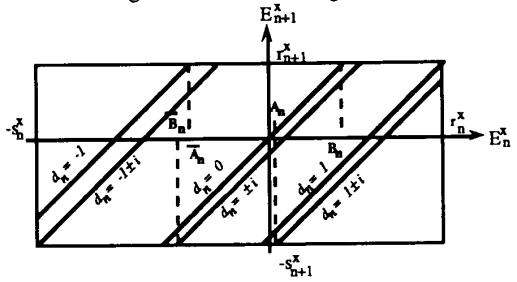

Fig. 2. *The Robertson Diagram for* E_n^x

The diagram is constructed as follows:

❑ we assume that E_n^x belongs to $[-s_n^x, r_n^x]$, which is the real part of R_n. s_n^x and r_n^x will be determined later.

❑ $E_{n+1}^x = E_n^x - \frac{1}{2}\ln\left[1+d_n^x 2^{-n+1} + \left(\left(d_n^x\right)^2 + \left(d_n^y\right)^2\right)2^{-2n}\right]$,

so the value of E_{n+1}^x vs. E_n^x is given by various straight lines parametrized by d_n^x and d_n^y.

❑ An adequate value of d_n^x is such that for any value of d_n^y (-1, 0 or +1), E_{n+1}^x "remains in the diagram" (i.e. $E_{n+1}^x \in [-s_{n+1}^x, r_{n+1}^x]$). Therefore r_{n+1}^x must be the largest value of

E^x_{n+1} corresponding to the straight line $d_n = 1$ of Fig. 2. That is, r^x_n must satisfy: $r^x_{n+1} = r^x_n - \ln(1+2^{-n})$. Since the length of R_n goes to 0 as n goes to $+\infty$, we deduce:

$$(4) \qquad r^x_n = \sum_{k=n}^{\infty} \ln\left(1+2^{-k}\right)$$

Similarly, the lowest possible value for E^x_n must correspond to the value obtained with $d_n = -1 \pm i$. This gives:

$$(5) \qquad s^x_n = -\frac{1}{2}\sum_{k=n}^{\infty} \ln\left(1-2^{-k+1}+2^{-2k+1}\right)$$

The terms $\bar{A}_n$, A_n, $\bar{B}_n$ and B_n appearing in Fig. 2 are:

$\bar{A}_n = r^x_{n+1} + \ln(1-2^{-n})$, $\bar{B}_n = -s^x_{n+1} + 2^{-1}(1+2^{-2n})$,

$A_n = -s^x_{n+1} + 2^{-1}\ln(1+2^{-n+1}+2^{-2n+1})$ and $B_n = r^x_{n+1}$

In the appendix, we have proven that $\bar{B}_n$ is less than $\bar{A}_n$, and that A_n is less than B_n. From this, for any $E^x_n \in [-s^x_n, r^x_n]$, the following choices give $E^x_{n+1} \in [-s^x_{n+1}, r^x_{n+1}]$:

$$(7) \qquad \begin{cases} \text{if } E^x_n < \bar{B}_n \text{ then } d^x_n = -1 \\ \text{if } \bar{B}_n \le E^x_n < \bar{A}_n \text{ then } d^x_n = -1 \text{ or } 0 \\ \text{if } \bar{A}_n \le E^x_n < A_n \text{ then } d^x_n = 0 \\ \text{if } A_n \le E^x_n \le B_n \text{ then } d^x_n = 0 \text{ or } 1 \\ \text{if } B_n < E^x_n \text{ then } d^x_n = +1 \end{cases}$$

2.b. Choice of d^y_n

Fig 3 shows the diagram assigned to the choice of d^y_n.

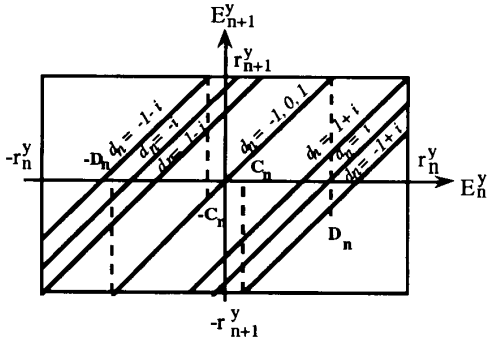

Fig. 3. *The Robertson Diagram for* E^y_n

As previously, we want our choice to be independent of the choice of d^x_n. From this, we deduce:

$$(8) \qquad r^y_n = \sum_{k=n}^{\infty} \tan^{-1}\left(\frac{2^{-k}}{1+2^{-k}}\right)$$

The terms C_n and D_n appearing in Fig. 3 are:

$$(9) \qquad C_n = -r^y_{n+1} + \tan^{-1}\left(\frac{2^{-n}}{1-2^{-n}}\right) \text{ and } D_n = r^y_{n+1}$$

In the appendix, we prove that C_n is less than D_n. Thus, for any E^y_n in $[-r^y_n, r^y_n]$, the following choices will hold:

$$(10) \qquad \begin{cases} \text{if } E^y_n < -D_n \text{ then } d^y_n = -1 \\ \text{if } -D_n \le E^y_n < -C_n \text{ then } d^y_n = -1 \text{ or } 0 \\ \text{if } -C_n \le E^y_n < C_n \text{ then } d^y_n = 0 \\ \text{if } C_n \le E^y_n \le D_n \text{ then } d^y_n = 0 \text{ or } 1 \\ \text{if } D_n < E^y_n \text{ then } d^y_n = +1 \end{cases}$$

Fig. 1 shows the rectangular domain R_1 where this algorithm gives a correct sequence d_n, inscribed in the domain A where such a sequence exists. The domain R_1 is:

$$[-0.829802\ldots, 0.868876\ldots] + i.[-0.749780\ldots, 0.749780\ldots]$$

2.c. The algorithm

Let us simplify the choice of d_n. Relations (7) and (10) involve comparisons that may require the examination of all the digits of the variables: we want to replace them by the examination of a small number of digits. We are going to find 3 constants $\bar{A}$, A and C whose fractional parts have a few digits only, say p_1 digits for $\bar{A}$ and A, and p_2 digits for C, such that for every n:

$$(11) \qquad \begin{cases} 2^n \bar{B}_n \le \bar{A} - 2^{-p1} < \bar{A} \le 2^n \bar{A}_n \\ 2^n A_n \le A \le A + 2^{-p1} \le 2^n B_n \\ 2^n C_n \le C \le C + 2^{-p2} \le 2^n C_n \end{cases}$$

Denote $\tilde{E}^x_n$ the number obtained by truncating $2^n E^x_n$ after its $p_1{}^{\text{th}}$ fractional digit, and $\tilde{E}^y_n$ the number obtained by truncating $2^n E^y_n$ after its $p_2{}^{\text{th}}$ fractional digit. We obtain, from (7), (10), (11):

- if $\tilde{E}^x_n \le \bar{A} - 2^{-p1}$ then $E^x_n \le \bar{A}_n$: we can choose $d^x_n = -1$
- if $\bar{A} \le \tilde{E}^x_n \le A$ then $\bar{B}_n \le E^x_n \le B_n$: we can choose $d^x_n = 0$
- if $A + 2^{-p1} \le \tilde{E}^x_n$ then $A_n \le E^x_n$: we can choose $d^x_n = 1$

(N.B. Since $\bar{A}$, A, and $\tilde{E}^x_n$ have at most p_1 fractional digits, if $\tilde{E}^x_n > \bar{A} - 2^{-p1}$, then $\tilde{E}^x_n \ge \bar{A}$).

- if $\tilde{E}^y_n \le -C - 2^{-p2}$ then $E^y_n \le -C_n$: we can choose $d^y_n = -1$
- if $-C \le \tilde{E}^y_n \le C$ then $-D_n \le E^y_n \le D_n$: we can choose $d^y_n = 0$
- if $C + 2^{-p2} \le \tilde{E}^y_n$ then $C_n \le E^y_n$: we can choose $d^y_n = 1$.

In the appendix, we have shown that $\bar{A} = -1/2$, $A = 1/4$, $C = 3/4$, $p_1 = 3$ and $p_2 = 4$ are convenient choices. This gives the E-mode of the BKM algorithm:

BKM Algorithm - E-mode

❑ Start with $E_1 \in R_1$ = [-0.829802...,+0.868876...] + i ·[-0.749780...,+0.749780...]

❑ Iterate: $\begin{cases} L_{n+1} = L_n\left(1+d_n 2^{-n}\right) \\ E_{n+1} = E_n - \ln\left(1+d_n 2^{-n}\right) \end{cases}$

with $d_n = d_n^x + i d_n^y$, (d_n^x, d_n^y = -1, 0, 1), chosen as follows: define $\tilde{E}_n^x$ as the number obtained by truncating the real part of $2^n E_n$ after its 3rd fractional digit, and $\tilde{E}_n^y$ as the number obtained by truncating the imaginary part of $2^n E_n$ after its 4th fractional digit.

$$\begin{cases} \text{if } \tilde{E}_n^x \le -5/8 \text{ then } d_n^x = -1 \\ \text{if } -1/2 \le \tilde{E}_n^x \le +1/4 \text{ then } d_n^x = 0 \\ \text{if } \tilde{E}_n^x \ge 3/8 \text{ then } d_n^x = +1 \end{cases} \quad \begin{cases} \text{if } \tilde{E}_n^y \le -13/16 \text{ then } d_n^y = -1 \\ \text{if } -3/4 \le \tilde{E}_n^y \le 3/4 \text{ then } d_n^y = 0 \\ \text{if } \tilde{E}_n^y \ge 13/16 \text{ then } d_n^y = +1 \end{cases}$$

❑ Result: $L_n \to L_1 e^{E_1}$

In practice, instead of computing E_n and examining the first digits of $\alpha_n = 2^n E_n$, one could directly compute the sequence $\alpha_{n+1} = 2\alpha_n - 2^{n+1}\ln(1+d_n 2^{-n})$.

2.d. Number of iterations

Let us estimate the number of iterations required to obtain a given accuracy. We want to compute $L_1 e^{E_1}$. The sequence d_i satisfies: $L_1 e^{E_1} = L_1 \prod_{i=1}^{\infty}\left(1+d_i 2^{-i}\right)$. After n iterations, we have computed $L_1 \prod_{i=1}^{n}\left(1+d_i 2^{-i}\right)$. The relative error made by approximating $L_1 e^{E_1}$ by this value is:

$$(12) \quad \left|\frac{L_1 e^{E_1} - L_1 \prod_{i=1}^{n}\left(1+d_i 2^{-i}\right)}{L_1 e^{E_1}}\right| = \left|1 - \frac{1}{\prod_{i=n+1}^{\infty}\left(1+d_i 2^{-i}\right)}\right|$$

One can show that this value is bounded by a term equivalent to 2^{-n}. Thus, *after* n *iterations of the E-mode of BKM, we obtain a relative error approximately equal to* 2^{-n}. So the error behaviour of BKM is the same as that of CORDIC.

2.e. Number of constants stored

This algorithm requires the pre computation and storage of:

- $\ln\left(1+d_i^x 2^{-i+1} + \left(d_i^{x2} + d_i^{y2}\right)2^{-2i}\right)$, d_i^x , d_i^y = -1, 0, 1
- $\tan^{-1}\left(\frac{2^{-i}}{1+d_i^x 2^{-i}}\right)$, d_i^x = -1, 0, 1

so, we need to store 8 terms for each value of i. From section 2.d, we deduce that, in order to obtain approximately n accuracy binary digits, we need to store $8n$ constants.

3. Computation of the complex logarithm function (L-mode)

Computing ln (L_1) using BKM requires the calculation of a sequence $d_n \in D$ = {-1,0,1,-i,i,i-1,i+1,-i-1,-i+1}, such that:

$$(13) \quad L_{n+1} = L_n\left(1 + d_n 2^{-n}\right) \to 1$$

Fig. 4 shows the set $B = \left\{\prod_{n=1}^{\infty}\left(1+d_n 2^{-n}\right)^{-1} \;\middle|\; d_n \in D\right\}$ of the numbers L_1 such that such a sequence d_n exists.

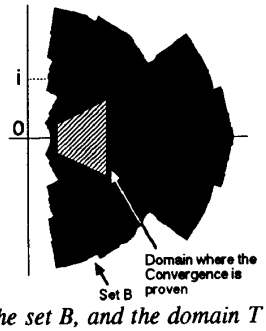

Fig. 4 *The set B, and the domain T where the convergence of the algorithm is proven.*

3.a A Straightforward strategy

In the following, we use the norm ||a+ib|| = max {|a|, |b|}. Define a sequence ε_n as: $\varepsilon_n = 2^n (L_n - 1)$. We obtain:

$$(14) \quad \varepsilon_{n+1} = 2(\varepsilon_n + d_n) + d_n \varepsilon_n 2^{-n+1}$$

If we find a sequence d_n such that the terms ε_n are bounded, then (13) will be satisfied. An intuitive solution is to choose d_n roughly equal to $-\varepsilon_n$. So, in this section, we consider the following strategy which gives $\|\varepsilon_n\| \le 3/2$:

- at step i, we examine the value $\tilde{\varepsilon}_i$ obtained by truncating the real and imaginary parts of ε_i after their p^{th} fractional digits, where p is a small integer.
- d_i is obtained by rounding the real and imaginary part of $-\tilde{\varepsilon}_i$ to the nearest integer. Since p is small, this is easily performed. If $\|\varepsilon_i\| \le 3/2$, this choice will give $d_i \in D$.

If this algorithm actually gives $\|\varepsilon_n\| \le 3/2$ for any n, then the sequence d_n will fulfill (13). From $\|\tilde{\varepsilon}_i - \varepsilon_i\| \le 2^{-p}$ and $\|d_i + \tilde{\varepsilon}_i\| \le 1/2$, using (14), we deduce: $\|\varepsilon_{n+1}\| \le 1 + 2^{1-p} + 2^{-n+1}\|d_n \varepsilon_n\|$. The norm $\|.\|$ satisfies $\|zz'\| \le 2\|z\|\cdot\|z'\|$, therefore:

$$(15) \quad \|\varepsilon_{n+1}\| \le 1 + 2^{1-p} + 2^{-n+2}\|\varepsilon_n\|$$

If $n \ge 4$, $p \ge 4$, and if $\|\varepsilon_4\| \le 3/2$, then, using (15), one can prove that for any $n \ge 3$, $\|\varepsilon_n\| \le 3/2$. Thus, if we start the iteration (14) at step 4, from ε_4 satisfying $\|\varepsilon_4\| \le 3/2$, then the strategy presented above will hold. This strategy allows computation of logarithms in a very tiny domain only: we can use it to compute ln (L_4) if $\|\varepsilon_4\| = \|16(L_4 - 1)\| \le 3/2$, i.e. if $L_4 \in$ [1-3/32,1+3/32] + i . [-3/32,+3/32].

3.b Computation in a larger domain

We still study the sequence $\varepsilon_k = 2^k (L_k - 1)$. Our purpose is to start its evaluation with ε_1 belonging to a domain that will be given later, and to obtain, after n steps ($n \geq 3$), a value ε_{n+1} such that $\| \varepsilon_{n+1} \| \leq 3/2$. After this, the strategy of section 3.a can be used. The following algorithm was found through simulations, before being proved.

BKM Algorithm - L-mode

❑ Start with L_1 belonging to the trapezoid T delimited by the straight lines $x = 1/2$, $x = 1.3$, $y = x/2$, $y = -x/2$.

❑ Iterate: $\begin{cases} L_{n+1} = L_n\left(1 + d_n 2^{-n}\right) \\ E_{n+1} = E_n - \ln\left(1 + d_n 2^{-n}\right) \end{cases}$

with $d_n = d_n^x + i d_n^y$, $(d_n^x, d_n^y = -1, 0, 1)$, chosen as follows:

- define ε_n^x and ε_n^y as the real and imaginary parts of $\varepsilon_n = 2^n (L_n - 1)$, and $\tilde{\varepsilon}_n^x$ and $\tilde{\varepsilon}_n^y$ as the values obtained by truncating these numbers after their 4^{th} fractional digits.
- At step 1:

$$\begin{cases} \text{if } \tilde{\varepsilon}_1^x \leq -7/16 \text{ and } 6/16 \leq \tilde{\varepsilon}_1^y \text{ then } d_1 = 1 - i \\ \text{if } \tilde{\varepsilon}_1^x \leq -7/16 \text{ and } \tilde{\varepsilon}_1^y \leq -6/16 \text{ then } d_1 = 1 + i \\ \text{if } -6/16 \leq \tilde{\varepsilon}_1^x \text{ and } 8/16 \leq \tilde{\varepsilon}_1^y \text{ then } d_1 = -i \\ \text{if } -6/16 \leq \tilde{\varepsilon}_1^x \text{ and } \tilde{\varepsilon}_1^y \leq -9/16 \text{ then } d_1 = i \\ \text{if } \tilde{\varepsilon}_1^x \leq -7/16 \text{ and } -5/16 \leq \tilde{\varepsilon}_1^y \leq 5/16 \text{ then } d_1 = 1 \\ \text{if } -6/16 \leq \tilde{\varepsilon}_1^x \text{ and } -1/2 \leq \tilde{\varepsilon}_1^y \leq 1/2 \text{ then } d_1 = 0 \end{cases}$$

- At step n, $n \geq 2$:

$$\begin{cases} \text{if } \tilde{\varepsilon}_n^x \leq -1/2 \text{ then } d_n^x = 1 \\ \text{if } -1/2 < \tilde{\varepsilon}_n^x < 1/2 \text{ then } d_n^x = 0 \\ \text{if } 1/2 \leq \tilde{\varepsilon}_n^x \text{ then } d_n^x = -1 \end{cases} \quad \begin{cases} \text{if } \tilde{\varepsilon}_n^y \leq -1/2 \text{ then } d_n^y = 1 \\ \text{if } -1/2 < \tilde{\varepsilon}_n^y < 1/2 \text{ then } d_n^y = 0 \\ \text{if } 1/2 \leq \tilde{\varepsilon}_n^y \text{ then } d_n^y = -1 \end{cases}$$

❑ Result: $E_n \to E_1 + \ln(L_1)$

In practice, instead of computing L_n, one could directly compute $\varepsilon_n = 2^n (L_n - 1)$ using (14).

Proof of the algorithm: our goal is to show that if $L_1 \in T$, then there exists $n \geq 4$ such that $\|\varepsilon_n\| \leq 3/2$. In order to do this, we build a sequence β_k of *bounding sets*, such that for any $L_1 \in T$, $\varepsilon_k \in \beta_k$. Our problem is reduced to show that there exists $n \geq 4$ such that β_n is included in the square $\|z\| \leq 3/2$. At the outset, let us explain how the sequence β_k is computed. Figures 5, and 6 show how β_{k+1} is deduced from β_k. The example described in these figures is imaginary: the "true" bounding sets are shown in Fig. 7.

❑ β_1 is equal to $2(T-1)$, and β_k is defined as an aggregate of convex polygons, represented by their vertices.

A step of the algorithm can be represented by a splitting of the complex plane into 9 convex *d-areas*. The d-area associated with $\delta \in D$ is the domain DA(δ) such that if $\tilde{\varepsilon}_k$ belongs to DA(δ), then the algorithm gives $d_k = \delta$. For instance, if $k \geq 2$, then DA(-1-i) is the set of the complex whose real and imaginary parts are greater than 1/2. In DA(δ), the transformation $\varepsilon_{k+1} = 2(\varepsilon_k + \delta) + \delta\varepsilon_k 2^{-k+1}$ is a *similarity*, i.e. the combination of a *rotation* and a multiplication by a real factor.

❑ Each convex polygon of β_k is splitted into sub-convex polygons, obtained by intersecting it with the d-areas. Fig. 5 shows the bounding set at step k, and the various d-areas ($k \geq 2$), and the splitting of the polygons of β_k.

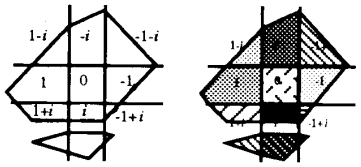

Fig. 5 **Left**: *The bounding set at step k and the different d-areas (for k ≥ 2)* **right**: *The bounding set is split into convex polygons following the d-areas*

Broadly speaking, β_{k+1} is obtained by computing the transformation of each sub-convex polygon generated by the splitting (the image of a polygon is obtained by computing the image of its vertices). We must take into account that d_k is deduced from $\tilde{\varepsilon}_k^x$ and $\tilde{\varepsilon}_k^y$, which are obtained by truncating the real and imaginary parts of ε_k after their 4^{th} fractional digits. For instance, if $\tilde{\varepsilon}_k = \tilde{\varepsilon}_k^x + i\,\tilde{\varepsilon}_k^y$ belongs to DA(-1), this does not prove that ε_k belongs to DA(-1). Thus, to each sub-convex polygon, a "ribbon" of length 2^{-4} is added, so that if $\tilde{\varepsilon_k}$ belongs to the "old" sub-polygon, then ε_k belongs to the "new" one. Then, for each new sub-polygon, we compute the image of its vertices by the similarity defined by the value of d_k assigned to the polygon (Fig. 6). This gives the new bounding set β_{k+1}.

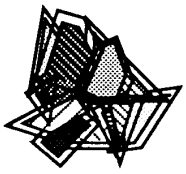

Fig. 6 *The iteration is applied to each of the vertices of the sub-polygons, to obtain the new bounding set*

The proof that $\varepsilon_k \in \beta_k$ for any $L_1 \in T$ is obvious. Thus, if we find $n \geq 4$ such that all the vertices of the sub-convex polygons of β_n are in the square $\|z\| \leq 3/2$, then the algorithm is proven. The adequate value of n is 6: this leads to a number of vertices much too large to be verified manually. We used a program written in ML for computing all the vertices of β_6, using exact rational arithmetic. Fig. 7 shows the bounding sets β_1, β_4 and β_6. Using this pro-

gram, we have verified that all the vertices of β_6 are included in the square $\|z\| \le 3/2$. Fig. 4 shows the domain T where the convergence of the algorithm is proven.

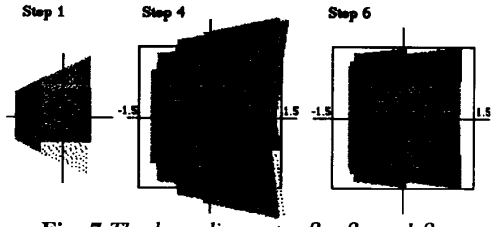

Fig. 7 *The bounding sets* β_1, β_4 *and* β_6.

3.c Number of iterations

As we did in section 2.d for the E-mode, let us estimate the number of iterations required to obtain a given accuracy. The sequence d_i defined by the algorithm satisfies:

$$\ln(L_1) = -\sum_{k=1}^{\infty} \ln\left(1 + d_k 2^{-k}\right)$$

after n iterations of the L-mode of BKM, we have computed $E_1 - \sum_{k=1}^{n} \ln\left(1 + d_k 2^{-k}\right)$. The absolute error made by approximating $E_1 + \ln(L_1)$ by this value is:

$$(16) \qquad error\ (n) = \left|\sum_{k=n+1}^{\infty} \ln\left(1 + d_k 2^{-k}\right)\right|$$

Using the Taylor expansion of the logarithm, one can show that this expression is bounded by a term equivalent to $2^{-n}\sqrt{2}$. Therefore, in order to obtain an absolute error less than 2^{-n}, one needs to perform $n+1$ iterations.

4. Application: computation of elementary functions

As shown in the previous sections, the BKM algorithm makes it possible to compute the following functions:

- in E-mode, $L_1 e^{E_1}$, where E_1 belongs to the domain [-0.829802,+0.868876] + i ·[-0.749780,+0.749780].
- in L-mode, $E_1 + \ln(L_1)$, where L_1 belongs to the trapezoid T delimited by the straight lines $x = 1/2$, $x = 1.3$, $y = \pm x/2$ (the actual convergence domain looks larger, but the algorithm is proven only for $L_1 \in T$).

Therefore, using BKM, one can compute the following functions of real variables:

4.a Functions computable using one mode of BKM.

❑ **real sine and cosine functions.** In the E-mode of BKM, one can compute the exponential of $E_1 = i\theta$ (where θ is a *real* number), and obtain $L_n = \cos\theta + i \sin\theta \pm 2^{-n}$.

❑ **real exponential function.** If E_1 is a real number belonging to [-0.829802...,+0.868876...], the E-mode of BKM will give a value L_n equal to $L_1 e^{E_1} \pm 2^{-n}$.

❑ **real logarithm.** If L_1 is a real number belonging to T, the E-mode of BKM will give $E_n = E_1 + \ln(L_1) \pm 2^{-n}$. Furthermore, in this case, the iteration is reduced to Brigg's algorithm, and the algorithm works for $L_1 \in \left[\prod_{n=1}^{\infty}\left(1+2^{-n}\right)^{-1}, \prod_{n=2}^{\infty}\left(1-2^{-n}\right)^{-1}\right] \approx [0.419, 1.731]$.

❑ **2-D rotations.** As pointed out in many papers dealing with CORDIC (e.g. [5]), performing rotations is useful for Fast Fourier Transformation, Digital Filtering, and Matrix Computations. The vector $(c\ d)^t$ obtained by rotating the 2-D vector $(a\ b)^t$ of an angle θ is computed using the E-mode of BKM, with $L_1 = a + ib$ and $E_1 = i\theta$.

❑ **real tan^{-1} function.** From the relation:

$$\ln(x+iy) = \begin{cases} \frac{1}{2}\ln\left(x^2+y^2\right) + i\tan^{-1}\frac{y}{x} \mod(2i\pi) & \text{if } x > 0 \\ \frac{1}{2}\ln\left(x^2+y^2\right) + i\left(\pi + \tan^{-1}\frac{y}{x}\right) \mod(2i\pi) & \text{if } x < 0 \end{cases}$$

one can deduce that, if $x + iy$ belongs to the convergence domain of the L-mode of BKM, then $\tan^{-1} y/x$ is the imaginary part of the limit value of E_n, while $0.5 \ln(x^2 + y^2)$ is its real part, assuming that the L-mode is used with $E_1 = 0$ and $L_1 = x + iy$.

4.b Functions computable using two consecutive modes of BKM.

❑ **Complex multiplication:** The product zt is evaluated as $z.e^{\log t}$, see fig. 8.

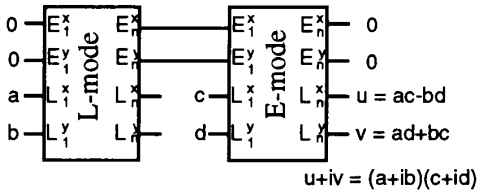

Fig. 8 *Complex multiplication*

In fact, one can compute zte^u, where z, t and u are complex numbers, using the same operator, by choosing E_1^x equal to the real part of u, and E_1^y equal to its imaginary part.

❑ **Computation of $x\sqrt{a}$ and $y\sqrt{a}$ in parallel** (x, y and a are real numbers): we use the relation $\sqrt{a} = e^{1/2 \ln a}$.

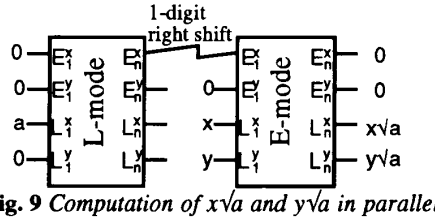

Fig. 9 *Computation of* $x\sqrt{a}$ *and* $y\sqrt{a}$ *in parallel*

❑ **Computation of lengths and normalization of 2D-vectors**: The L-mode of BKM allows the computation of $F = 1/2 \ln(a^2+b^2) = \ln\sqrt{a^2+b^2}$, where a and b are real numbers. Using the E-mode of BKM, we can compute e^F, or e^{-F}. See Fig. 10. The normalization of 2D vectors (i.e. the computation of $x/\sqrt{a^2+b^2}$ and $y/\sqrt{a^2+b^2}$, where x, y, a and b are real numbers) is a basic step of Givens' QR factorization algorithm.

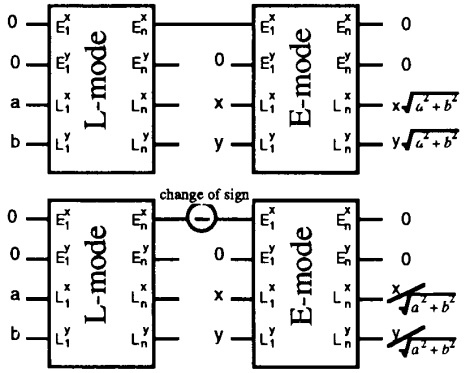

Fig. 10 *Lengths and normalization of 2D-vectors*

5. Comparison with CORDIC

To obtain p significant bits, CORDIC and BKM roughly need p iterations. BKM requires the storage of $8p$ constants, while CORDIC requires the storage of p constants. Since these constants are represented by p-digits, both algorithms need a $O(p^2)$ area for storage of them. Both algorithms need a shifter able to perform an n-position shift at step n. A barrel shifter makes it possible to perform a n-position shift (for any $n \le p$) in constant time, and lies in an area $O(n^2)$. Since the area complexity of most adders is better than $O(n^2)$, we deduce that the area complexity of CORDIC and BKM is $O(n^2)$. The computations performed during a BKM iteration are:

● **For the variable E_n:**

$$\begin{cases} E^x_{n+1} = E^x_n - \frac{1}{2}\ln\left[1 + d^x_n 2^{-n+1} + \left(d^{x2}_n + d^{y2}_n\right)2^{-2n}\right] \\ E^y_{n+1} = E^y_n - d^y_n \tan^{-1}\left(\frac{2^{-n}}{1 + d^x_n 2^{-n}}\right) \end{cases}$$

or

$$\begin{cases} \alpha^x_{n+1} = 2\alpha^x_n - 2^n \ln\left[1 + d^x_n 2^{-n+1} + \left(d^{x2}_n + d^{y2}_n\right)2^{-2n}\right] \\ \alpha^y_{n+1} = 2\alpha^y_n - 2^{n+1} d^y_n \tan^{-1}\left(\frac{2^{-n}}{1 + d^x_n 2^{-n}}\right) \end{cases}$$

if instead of computing E_n and examining the first digits of $\alpha_n = 2^n E_n$, we directly compute α_n.

● **For the variable L_n:**

$$\begin{cases} L^x_{n+1} = L^x_n + \left(d^x_n L^x_n - d^y_n L^y_n\right)2^{-n} \\ L^y_{n+1} = L^y_n + \left(d^y_n L^x_n + d^x_n L^y_n\right)2^{-n} \end{cases}$$

or

$$\begin{cases} \varepsilon^x_{n+1} = 2\left(\varepsilon^x_n + d^x_n\right) + \left(d^x_n \varepsilon^x_n - d^y_n \varepsilon^y_n\right)2^{-n+1} \\ \varepsilon^y_{n+1} = 2\left(\varepsilon^y_n + d^y_n\right) + \left(d^x_n \varepsilon^y_n + d^y_n \varepsilon^x_n\right)2^{-n+1} \end{cases}$$

if instead of computing L_n and examining the first digits of $\varepsilon_n = 2^n (L_n - 1)$, we directly compute ε_n.

So the BKM iterations look more complicated than the CORDIC iterations. As a matter of fact, in order to compare CORDIC and BKM, we have to assume that we use a redundant number system. Using such a system, the time complexities of both algorithms are $O(p)$. As pointed out in many papers dealing with CORDIC, efficient use of CORDIC with such a number system requires a doubling of the iterations in space [3] or in time [10]. For instance, doubling the CORDIC iterations in time gives:

$$\begin{cases} x_{n+1} = x_n - d_n y_n 2^{-n} - d^2_n x_n 2^{-2n-2} \\ y_{n+1} = y_n + d_n x_n 2^{-n} - d^2_n y_n 2^{-2n-2} \\ z_{n+1} = z_n - 2d_n \tan^{-1} 2^{-n-1} \end{cases}$$

which is at least as complex as the BKM iteration (because at step n one needs to perform an n position shift and a $2n$-2 position shift: this requires a larger shifter). Doubling the iterations in space requires more control: in the *branching CORDIC* method proposed by Duprat and Muller [3], one needs to compare at each step the values given by two CORDIC modules. Furthermore, doubling the iterations makes it possible to obtain a constant scaling factor, but this factor remains different from 1, therefore, for computing many functions, one needs to perform a multiplication after the CORDIC iterations. So, although both methods have the same time and space complexities, BKM looks more interesting when using a redundant number system.

6. Conclusion

We have proposed a new algorithm for the computation of many elementary functions (complex exponential and logarithms, complex multiplication, real functions sin, cos, $\tan^{-1} y/x$, $\ln(x^2+y^2)$, $x\sqrt{a}$, $x\sqrt{a^2+b^2}$, $x/\sqrt{a^2+b^2}$, 2D rotations). This algorithm matches the CORDIC algorithm, since it allows the use of a redundant number system without any scaling factor problem. Moreover, several functions (complex exponentials and logarithms, complex multiplications), are directly computable using one or two BKM operations, while this is not true using CORDIC.

References

[1] A. Avizienis, *Signed-digit number representations for fast parallel arithmetic*, IRE Transactions on electronic computers, 10, pp. 389-400, 1961.

[2] T.C. Chen, Automatic Computation of Exponentials, Logarithms, Ratios and Square Roots, IBM J. Res. Develop, 16, pp 380-388, 1972.

[3] J. Duprat and J.M. Muller, *The CORDIC Algorithm: new results for fast VLSI implementation*, to appear in IEEE Trans. Computers.

[4] M.D. Ercegovac, *A general hardware-oriented method for evaluation of functions and computations in a digital computer*, IEEE Transactions on Computers, Vol. C-26 No 7, July 1977, pp 667-680.

[5] Y.H. Hu, CORDIC-based VLSI Architectures for Digital Signal Processing, IEEE Signal Processing Magazine, pp 16-35, July 1992.

[6] K. Hwang, *Computer Arithmetic principles, architecture and design*, J. Wiley&Sons Inc., New-York, 1979.

[7] B. De Lugish, *A class of algorithms for automatic evaluation of functions and computations in a digital computer*, PhD dissertation, Department of Computer Science, Univ. of Illinois, Urbana, June 1970.

[8] J.E. Robertson, *A new class of digital division methods*, IRE Transactions on Electronic Computers, Vol. EC-17, Sept. 1958.

[9] H.E. Salzer, *radix tables for finding the logarithm of any number of 25 decimal places*, in *Tables of functions and of zeros of functions*, National Bureau of Standards Applied Mathematics Series No 37, 1954, pp 143-144.

[10] N. Takagi, T. Asada and S. Yajima, *Redundant CORDIC methods with a constant scale factor*, IEEE Trans. on Computers, Vol. 40 No 9, pp 989-995, Sept. 1991.

[11] J. Volder, *The CORDIC Computing technique*, IRE Transactions on Computers, Sept. 1959.

[12] J. Walther, *A unified algorithm for elementary functions*, Joint Computer Conf. proc., Vol. 38, 1971.

Appendix

Computation of the parameters occurring in the E-mode.

We want to prove that $\bar{B}_n < \bar{A}_n$, $A_n < B_n$, $C_n < D_n$, and to find two numbers $\bar{A}$ and A whose binary representations have only p_1 fractional digits, and a p_2 fractional digit number C, such that for any n:

$$2^n \bar{B}_n \le \bar{A} - 2^{-p_1} \le \bar{A} \le 2^n \bar{A}_n$$
$$2^n A_n \le A \le A + 2^{-p_1} \le 2^n B_n$$
$$2^n C_n \le C \le C + 2^{-p_2} \le 2^n D_n$$

with:

$$\bar{A}_n = \ln\left(1-2^{-n}\right) + \sum_{k=n+1}^{\infty} \ln\left(1+2^{-k}\right)$$

$$A_n = \frac{1}{2}\ln\left(1+2^{-n+1}+2^{-2n+1}\right) + \frac{1}{2}\sum_{k=n+1}^{\infty} \ln\left(1-2^{-k+1}+2^{-2k+1}\right)$$

$$\bar{B}_n = \frac{1}{2}\ln\left(1+2^{-2n}\right) + \frac{1}{2}\sum_{k=n+1}^{\infty} \ln\left(1-2^{-k+1}+2^{-2k+1}\right)$$

$$B_n = \sum_{k=n+1}^{\infty} \ln\left(1+2^{-k}\right)$$

$$C_n = \tan^{-1}\left(\frac{2^{-n}}{1-2^{-n}}\right) - \sum_{k=n+1}^{\infty} \tan^{-1}\left(\frac{2^{-k}}{1+2^{-k}}\right)$$

$$D_n = \sum_{k=n+1}^{\infty} \tan^{-1}\left(\frac{2^{-k}}{1+2^{-k}}\right)$$

For $n = 1$, we easily obtain:

$$2\bar{B}_1 \le -2^{-1} - 2^{-3} \le -2^{-1} \le 2\bar{A}_1$$
$$2A_1 \le 2^{-2} \le 2^{-1} \le 2B_1$$
$$2C_1 \le 2^{-1} + 2^{-2} \le 2^{-1} + 2^{-2} + 2^{-4} \le 2^n C_n$$

Using Taylor expansions, we get the following bounds:

$$2^n \bar{A}_n \ge -\frac{2}{3}2^{-n} - \frac{2}{3}2^{-2n} \ge -\frac{1}{4}$$
$$2^n \bar{B}_n \le -1 + \frac{1}{2}2^{-n} + \frac{4}{21}2^{-2n} \le -\frac{1}{2}$$
$$2^n B_n \ge 1 - \frac{1}{6}2^{-n} \ge \frac{1}{2}$$
$$2^n A_n \le \frac{2}{21}2^{-n} \le \frac{1}{4}$$
$$2^n D_n \ge 1 - \frac{1}{3}2^{-n} \ge \frac{1}{2} + \frac{1}{4} + \frac{1}{16}$$
$$2^n C_n \le \frac{4}{3}2^{-n} + \frac{2}{3}2^{-2n} \le \frac{1}{2}$$

These relations and the relations obtained for $n = 1$ give:

$$2^n \bar{B}_n \le -\frac{1}{2} - \frac{1}{8} \le -\frac{1}{2} \le 2^n \bar{A}_n$$
$$2^n A_n \le \frac{1}{4} \le \frac{1}{4} + \frac{1}{8} \le 2^n B_n$$
$$2^n C_n \le \frac{1}{2} + \frac{1}{4} \le \frac{1}{2} + \frac{1}{4} + \frac{1}{16} \le 2^n D_n$$

From this, we deduce that $\bar{B}_n < \bar{A}_n$, $A_n < B_n$, $C_n < D_n$, and that the following parameters fulfill the requirements presented above:

$$\bar{A} = \frac{1}{2} \qquad A = \frac{1}{4} \qquad C = \frac{1}{2} + \frac{1}{4}$$
$$p_1 = 3 \qquad p_2 = 4$$