High Radix BKM algorithm with Selection by Rounding

Laurent-Stéphane Didier, Fabien Rico
Laboratoire d'Informatique de Paris 6

January 21, 2002

Abstract

We present in this paper a high radix implementation of BKM algorithm. This is a shift and add CORDIC-Like algorithm that allows fast computations of complex exponential and logarithm. The improvement lies in fewer iterations for a given precision and in the reduction of the size of lookup tables for high radices.

Keywords: Elementary function, CORDIC algorithm, Computer Arithmetic, High Radix

1 Introduction

Several class of algorithm for computing complex exponential and logarithm have been proposed. A first class is composed by algorithms using polynomial approximation of the function to be computed [Tan89, Smi89, Mul97, DMF00]. Generally, few iterations have to be done by this kind of algorithm, but multiplications are required. Next, multipartite tables methods which can be implemented for low precision require very few arithmetic operators [SM95, HT95, dDT00]. On the opposite, quadratic convergence algorithm allow better precision, but are complex and uneasy to implement [Bre76, BB84]. Such algorithms have the best asymptotic complexity, but become really efficient only for thousands bits. Finally, shift and add algorithms, which most famous is CORDIC [Vol59, Wal71, DM96], allow fairly good precision (until hundreds bits) with low cost iterations [Mul97]. Similarly to CORDIC algorithm, BKM algorithm allows the evaluation of many elementary functions [BKM94, BI99]. In order to reduce the number of iterations for a given precision, it was proposed for CORDIC to use high radix implementation [ALB00a, ALB00b, Lew99].

This paper presents a high radix implementation of the BKM algorithm for computing complex logarithm and exponential without scaling factor. We show a method for choosing the digits and a convergence domain for BKM algorithm for high radix. A first method using the radix $\beta = 10$ as been presented in [IMR00] but this method spend a lot of memory. Indeed, such an algorithm need a number of digits of the order of magnitude of β^2. As each iteration will need to store a complex number in a table by digits, it will use an $O(N \times \beta^2)$ entry table (where N is final number of iterations) while CORDIC algorithm only use $O(N)$ entry table (see [ALB00a] and [ALB00b]).

The paper is organized as follows, we first present briefly the initial version of BKM algorithm. Next, we propose a modification of the itration used in this algorithm, designed to solve the principal problem of high-radix BKM which is the memory cost. Thereafter, we examine the two mode off BKM E-mode (section 3) for computing Complex exponential and L-mode (section 4) for complex logarithm. each section present the selection digits method, the domain of convergence and an example of argument reduction to this domain. Finally, section 5 we compare our algorithm with the other high radix CORDIC implementation and conclude.

1.1 Notations

Let consider a complex number z. We note its real part z^x and its imaginary z^y. Thus

$$z = z^x + iz^y$$

We call **Gauss numbers** the complex numbers whose real and imaginary part are integers:

$$d = d^x + id^y \quad \text{with } d^x, d^y \in \mathbb{Z}$$

We note $\lceil z \rfloor_p$ the value of the real and imaginary parts of z *rounded to the nearest number* with p fractional digits. For instance, we note $\lceil z \rfloor_0$ the gauss number closest to z.

We note $\langle z \rangle_p$ the value of the real and imaginary parts z *truncated down* with p fractional digits.

1.2 The BKM algorithm

The BKM algorithm presented in [BKM94] computes the following iterations:

$$\left\{ \begin{array}{lcl} E_{n+1} & = & E_n(1 + d_n\beta^{-n}) \\ L_{n+1} & = & L_n - \ln\left(1 + d_n\beta^{-n}\right) \end{array} \right.$$

where β is an integer and the values d_n are chosen so that:

$$\begin{gathered} d_n = d_n^x + id_n^y \\ d_n^x,\, d_n^y \in \{-a, -a+1, \ldots, 0, \ldots, a-1, a\} \quad \text{with} \quad a = \frac{\beta}{2} + 1 \end{gathered} \tag{1}$$

In this paper, the d_n values are called digits and the integer β is the radix of the algorithm.

In order to compute one iteration, the values $\ln\left(1 + d_n\beta^{-n}\right)$ are stored in a lookup table and are acceded by d_n. The size of this table directly depends on the number of possible digits. Thus, each iteration only needs shift, addition and multiplication by a digit d_n.

It is shown in [Mul97] that these iterations preserve the following equalities:

$$\frac{E_n}{E_1} = \frac{\exp(L_1)}{\exp(L_n)} \quad \text{and} \quad L_n - L_1 = \ln(E_1) - \ln(E_n)$$

As a consequence, it is possible to use this algorithm trough two different modes:

- **E-mode**: if we choose d_n so that L_n converges to 0, then E_n will converge to $E_1 e^{L_1}$ making possible the computation of the complex exponential
- **L-mode**: similarly, if we choose d_n so that E_n converges to 1, then L_n will converge to $L_1 + \ln(E_1)$ which permits the computation of the complex logarithm.

Initially this algorithm was developed in radix 2, simplifying the choice of digits d_n. However, this algorithm gives one digit of the result at each iteration. For instance, a radix 2 algorithm gives only one bit of the result at each step. As a consequence, a high radix implementation of the BKM algorithm would need less iterations for the same precision.

Unfortunately, three problems appear: the complexity of the digit selection, the numerical instability of the first iterations and the size of tables for high radix implementations.

It has been shown in [IMR00] that for radix 10 the BKM algorithm looks for digits in a set of one hundred elements. This means that each iteration requires a lookup table having one hundred entries, making this implementation inefficient for higher radices. Therefore, we will show a method for reducing the tables size.

2 Reducing the size of lookup tables

In order to minimize the set of digits, we propose similarly to [Lew99] to split each iteration in two half-iterations. The first half-iteration is designed to reduce the imaginary part. This iteration is close to the CORDIC iteration, see [Mul97]:

$$\left\{ \begin{array}{lcl} \overline{E}_n & = & E_n(1 + id_n^y\beta^{-n}) \\ \overline{L}_n & = & L_n - \ln\left(1 + id_n^y\beta^{-n}\right) \end{array} \right. \tag{2}$$

The second half-iteration has to reduce the real part. This computation is close to the BKM iteration [Mul97]):

$$\left\{ \begin{array}{lcl} E_{n+1} & = & \overline{E}_n(1 + d_n^x\beta^{-n}) \\ L_{n+1} & = & \overline{L}_n - \ln\left(1 + d_n^x\beta^{-n}\right) \end{array} \right. \tag{3}$$

Consequently, we can deduce from these iterations that for the E-mode (respectively the L-mode):

- the digits d_n^y are chosen in order to only minimize $\left|\overline{L}_n^y\right|$ (respectively $\overline{E}_n^y$),

- the digits d_n^x are chosen in order to only minimize $|L_n^x|$ (respectively E_n^x).

In this way, the number of iteration is doubled compared to the initial algorithm, but the number of digits is reduced by one order of magnitude because the digits are chosen from a smaller set:

$$\{-a, -a+1, \cdots, 0, \cdots, a-1, a\} \quad \cup \quad \{-ia, -i(a-1), \cdots, 0, \cdots, i(a-1), ia\} \quad \text{with} \quad a = \frac{\beta}{2} + 1$$

In other words, there are only $O(\beta)$ possible digits instead of $O(\beta^2)$.

Now, we will show that on a small convergence domain, it is possible to choose d_n^x or d_n^y only by rounding the value E_n (see 4.1) or L_n (see 3.1) which is more efficient than choosing a table of value as in [Lew99]. But in order to have a functional algorithm, it is necessary to extend the convergence domain by examining closely the two first iterations of our algorithm. To this end, we will study separately the E-mode and L-mode.

3 The computation of complex exponential : E-mode

In this mode, with regard to L_1, we construct a sequence of d_n^x, d_n^y such that the sequence L_n converges to 0. This section outlines a method for choosing the digits d_n^x and d_n^y. This choice is critical for the convergence and the performance of our algorithm. Indeed, a complex choice will lead to a rapid convergence on a wide domain but will be costly not only in term of computing time but also in term of hardware resources.

Because the set of possible digits is finite, we cannot always take the best value for d_n^x and d_n^y and we have to select an approximation which is relatively close to the best value. Unfortunately, this will not assure the convergence of the algorithm for the whole complex plane, but only for a small domain.

First, we present a method for choosing the digits d_n^x and d_n^y. Next, we will prove that, for any value taken from a defined domain, an iteration will keep the result in this domain. Thereafter, we will show an argument reduction from any part of the complex plane to this domain, allowing our algorithm to converge for any input.

3.1 Digit selection

Separating the real and imaginary part in equation 2 and 3 gives the two following iterations:

$$\begin{array}{rcl} \overline{L}_n^x & = & L_n^x - \frac{1}{2}\ln\left(1 + {d_n^y}^2 \beta^{-2n}\right) \\ \overline{L}_n^y & = & L_n^y - \arctan\left(d_n^y \beta^{-n}\right) \end{array} \quad \text{and} \quad \begin{array}{rcl} L_{n+1}^x & = & \overline{L}_n^x - \ln\left(1 + d_n^x \beta^{-n}\right) \\ L_{n+1}^y & = & \overline{L}_n^y \end{array} \tag{4}$$

The exact value for d_n^x and d_n^y that minimise L_{n+1}^x and L_{n+1}^y are :

$$d_n^x = \left(\exp\left(\overline{L}_n^x\right) - 1\right)\beta^n \quad \text{and} \quad d_n^y = \tan\left(L_n^y\right)\beta^n \tag{5}$$

With these digits, $L_{n+1}^x = L_{n+1}^y = 0$.But these are not integer values except if $\overline{L}_n^x = 0$ or $L_n^y = 0$. So we have to choose an integer close to these values. As L_n^x and L_n^y tend to 0, we can use an approximation of the exact functions in (5). Indeed :

$$\left(\exp\left(\overline{L}_n^x\right) - 1\right)\beta^n \simeq \overline{L}_n^x \beta^n \quad \text{and} \quad \tan\left(L_n^y\right)\beta^n \simeq L_n^y \beta^n \tag{6}$$

Let note

$$T_n = L_n \beta^n \quad \text{and} \quad \overline{T}_n = \overline{L}_n \beta^n \tag{7}$$

Thus, d_n^x and d_n^y respectively should be close to $\overline{T}_n^x$ and T_n^y. In order to obtain the integers d_n^x and d_n^y, we should round $\overline{T}_n^x$ and T_n^y to the nearest integer. This exact rounding implies exact comparisons which may be inefficient, if for instance, $\overline{T}_n^x$ and T_n^y are coded in a redundant number representation. Therefore, we will round the truncation of $\overline{T}_n^x$ and T_n^y keeping only 2 fractional digits:

$$\begin{array}{rcl} d_n^y & = & \left\lceil \langle T_n^y \rangle_2 \right\rfloor_0 \\ d_n^x & = & \left\lceil \langle \overline{T}_n^x \rangle_2 \right\rfloor_0 \end{array} \tag{8}$$

Such a choice is the result of several approximation and we must study the convergence of the algorithm. For this purpose, we study the evolution of the sequence T_n. Clearly, if it's possible to prove that T_n is bounded, then, L_n will tend to 0 and each iteration will reduce the error by a factor of β.

3.2 Correctness of the approximation

Suppose that the sequence $|T_n^x|$ is bounded by a value A. Then $|L_n^x|, |L_n^y| \leq A\beta^{-n}$. This means that each iteration should reduce the value of the sequences L_n^x and L_n^y by a factor β. We will show that if we use our digit selection method, this property is true except for the two first iterations.

First, we have to bound the value of T_{n+1} by a value depending of T_n. Let $a = \lceil T_n^x \rceil$, $b = \lceil T_n^y \rceil$. We have :

$$\begin{array}{rcl} |T_n^x| & \leq & a \\ |T_n^y| & \leq & b \end{array}$$

With the choice of d_n^x and d_n^y we obtain:

$$\begin{array}{rcl} \left|\langle \overline{T}_n^x \rangle_2 - T_n^x\right| & \leq & \beta^{-2} \\ \left|\langle T_n^y \rangle_2 - T_n^y\right| & \leq & \beta^{-2} \end{array}$$

and

$$\begin{array}{rcl} \left|d_n^x - \lceil \overline{T}_n^x \rfloor_0\right| & \leq & \frac{1}{2} \\ \left|d_n^y - \lceil T_n^y \rfloor_0\right| & \leq & \frac{1}{2} \end{array}$$

then

$$\begin{array}{rcl} \left|d_n^x - \overline{T}_n^x\right| & \leq & \frac{1}{2} + \beta^{-2} \\ \left|d_n^y - T_n^y\right| & \leq & \frac{1}{2} + \beta^{-2} \end{array}$$

It is well known that the function ln and arctan have the following properties:

- if $x \in [0, \frac{1}{2}]$ then

$$\left\{ \begin{array}{rcccl} x - \frac{x^2}{2} & \leq & \ln(1+x) & \leq & x \\ x - \frac{x^3}{3} & \leq & \arctan(x) & \leq & x, \end{array} \right.$$

- if $x \in [-\frac{1}{2}, 0]$ then

$$\left\{ \begin{array}{rcccl} x - x^2 & \leq & \ln(1+x) & \leq & x \\ x & \leq & \arctan(x) & \leq & x - \frac{x^3}{3}. \end{array} \right.$$

By definition of d_n^y and b, $|d_n^y| \leq b$. The first half-iteration can be written

$$\left[\begin{array}{rcl} \overline{T}_n^x & = & T_n^x - \frac{1}{2}\ln\left(1 + {d_n^y}^2 \beta^{-2n}\right)\beta^n \\ \overline{T}_n^y & = & T_n^y - d_n^y + d_n^y - \arctan\left(d_n^y \beta^{-n}\right)\beta^n, \end{array} \right.$$

which gives the following inequalities:

$$\begin{array}{rcccl} -a - \frac{b^2}{2}\beta^{-n} & \leq & \overline{T}_n^x & \leq & a \\ -\frac{1}{2} - \beta^{-2} - \frac{b^3}{3}\beta^{-2n} & \leq & \overline{T}_n^y & \leq & \frac{1}{2} + \beta^{-2} + \frac{b^3}{3}\beta^{-2n} \end{array}$$

By definition of d_n^x and a (which is an integer), we have : $-a - \frac{b^2}{2}\beta^{-n} - \frac{1}{2} \leq d_n^x \leq a$. The second half iteration is :

$$\left[\begin{array}{rcl} \frac{T_{n+1}^x}{\beta} & = & \overline{T}_n^x - d_n^x + d_n^x - \ln\left(1 + d_n^x \beta^{-n}\right)\beta^n \\ \frac{T_{n+1}^y}{\beta} & = & \overline{T}_n^y, \end{array} \right.$$

and gives the following bound:

$$\begin{array}{rrcccl} & -\frac{\beta}{2} - \beta^{-1} & \leq & T_{n+1}^x & \leq & \frac{\beta}{2} + \beta^{-1} + \left(a + \frac{b^2}{2}\beta^{-n} + \frac{1}{2}\right)^2 \beta^{-n+1} \\ \text{and} & -\frac{\beta}{2} - \beta^{-1} - \frac{b^3}{3}\beta^{-2n+1} & \leq & T_{n+1}^y & \leq & \frac{\beta}{2} + \beta^{-1} + \frac{b^3}{3}\beta^{-2n+1} \end{array} \qquad (9)$$

Now, let simplify the bounds of T_n. If $\beta \geq 10$ and:

- if $n=2$, $a=2$ and $b=\beta$ then $|T_3^x|$ and $|T_3^y| \leq \frac{\beta}{2}+1$,
- if $n \geq 3$, $a=b=\frac{\beta}{2}+1$ then $|T_{n+1}^x|$ and $|T_{n+1}^y| \leq \frac{\beta}{2}+1$.

By induction, this proves that the algorithm converges if $n \geq 2$ on the following domain:

$$P_E = \left[1-\frac{2}{\beta^2}, 1+\frac{2}{\beta^2}\right] + i\left[-\frac{1}{\beta}, \frac{1}{\beta}\right] \tag{10}$$

3.3 Argument reduction

Because of the method for constructing d_n^x and d_n^y this domain remain very small. Indeed, the best choice for these numbers (see section 3.1) are numbers minimizing the values:

$$\left|\overline{T}_n^x\beta^{-n} - \ln\left(1+d_n^x\beta^{-n}\right)\right| \quad \text{and} \quad \left|T_n^y\beta^{-n} - \arctan\left(1+d_n^y\beta^{-n}\right)\right| \tag{11}$$

The proposed choice (eq. 8) is efficient only if (see 6):

$$\left(\exp\left(\overline{L}_n^x\right)-1\right)\beta^n \simeq \overline{L}_n^x\beta^n \qquad \text{and} \qquad \tan\left(L_n^y\right)\beta^n \simeq L_n^y\beta^n$$

Unfortunately, this is not true for the two first iterations. In other words, if $n=1$ or $n=2$ then T_{n+1}^x and T_{n+1}^y may be greater than $\frac{\beta}{2}+1$. So, these iterations have to be replaced. Several methods can be used for the first iterations:

- It is possible to choose the values d_n^x and d_n^y from a lookup table indexed by the most significant bits of $\overline{T}_n^x$ and T_n^y. This allows a better choice for the digits and will improve the reduction.
- It is possible to double an iteration that will also improve the reduction but will need more computations.
- It is possible to change the radix of the algorithm, computing an iteration with radix 4β for instance. This is equivalent to code the fractional part of d_n^x and d_n^y with 2 bits. The number of possible digits increases, but the choice and the reduction are better.
- It is possible to enlarge the set of possible digits for the next iteration.
- It is also possible to simulate the two first iteration by several iterations of radix 2 BKM algorithm.

Example: Suppose that $\beta \geq 10$. The first iterations can be computed as follow:
Fist, note we will start from the following domain:

$$\mathcal{D} = [\ln(2), 2\ln(2)] + i\left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$$

Reducing to this domain is well known as shown in [Mul97]. We have to improve the reduction from $\mathcal{D}$ to the domain P_E (see 10). This second reduction will only use two iteration of BKM, but those iterations are performed differently. The two first iterations are computed as follow:

- For the first iteration, d_n^y and d_n^x are chosen from two lookup tables indexed by T_n^y and respectively $\overline{T}_n^y$ rounded with 1 bit for the fractional part.
- The set of digits for the second iteration is extended to $\{-\beta, \ldots, \beta\}$. Furthermore, this iteration is performed twice.

More precisely, we define $\Delta^x(n)$ and $\Delta^y(n)$ being two lookup tables such that:

- $\Delta^x(n)$ is the integer d minimizing the value $\left|\frac{n}{2\beta} - \ln\left(1+d\beta^{-1}\right)\right|$ for $0 \leq n \leq \lceil 4\beta\ln(2)\rceil$.
- $\Delta^y(n)$ is the integer d minimizing the value $\left|\frac{n}{2\beta} - \arctan\left(d\beta^{-1}\right)\right|$ for $-2\beta \leq n \leq 2\beta$.

These tables verify:

$$\left|\frac{n}{2\beta} - \ln\left(1+\Delta^x(n)\beta^{-1}\right)\right| \leq \frac{1}{2\beta} \tag{12a}$$

$$\left|\frac{n}{2\beta} - \arctan\left(\Delta^y(n)\beta^{-1}\right)\right| \leq \frac{1}{2\beta} \tag{12b}$$

We can do the iterations given in table 1.

Step 1	$\left\{\begin{array}{lcl} d_1^y & = & \Delta^y(\lceil 2\langle T_1^y\rangle_2\rfloor_0) \\ \overline{L}_1 & = & L_1 - \ln\left(1+id_1^y\beta^{-1}\right)\end{array}\right.$
Step $1+\frac{1}{2}$	$\left\{\begin{array}{lcl} d_1^x & = & \Delta^x(\left\lceil 2\left\langle \overline{T}_1^x\right\rangle_2\right\rfloor_0) \\ L_2 & = & \overline{L}_1 - \ln\left(1+d_1^x\beta^{-1}\right)\end{array}\right.$
Step 2	$\left\{\begin{array}{lcl} d_2^y & = & \lceil\langle T_2^y\rangle_2\rfloor_0 \\ \overline{L}_2 & = & L_2 - \ln\left(1+id_2^y\beta^{-2}\right)\end{array}\right.$
Step $2+\frac{1}{2}$	$\left\{\begin{array}{lcl} d_2^x & = & \left\lceil\left\langle \overline{T}_2^y\right\rangle_2\right\rfloor_0 \\ L & = & \overline{L}_2 - \ln\left(1+d_2^x\beta^{-2}\right)\end{array}\right.$

Table 1: The 4 half-iteration used by the E-mode

Considering $L_1 \in \mathcal{D}$, we have:

$$\begin{array}{rcccl} 0 & \leq & L_1^x & \leq & \ln(2) \\ -\frac{\pi}{4} & \leq & L_1^y & \leq & \frac{\pi}{4}\end{array}$$

By definition of T_1^y (see 7), $|T_1^y| < \beta$ then $\left|\lceil 2\langle T_1^y\rangle_2\rfloor_0\right| \leq 2\beta$ and the value $d_1^y = \Delta^y(\lceil 2\langle T_1^y\rangle_2\rfloor_0)$ exists in the table.

We do the following iteration:

$$\overline{L}_1^x = L_1^x - \frac{1}{2}\ln\left(1+(d_1^y\beta^{-1})^2\right) \quad (13a)$$

$$\overline{L}_1^y = L_1^y - \arctan\left(d_1^y\beta^{-1}\right) \quad (13b)$$

By definition of d_1^y, we have $|d_1^y| \leq \beta$ and with equation (13a), it easily gives a bound for $\overline{L}_1^x$:

$$\begin{array}{rcccl} \ln(2) - \frac{1}{2}\ln(2) & \leq & \overline{L}_1^x & \leq & 2\ln(2) \\ 0 & \leq & \overline{L}_1^x & \leq & 2\ln(2)\end{array} \quad (14)$$

Let us note :

$$\begin{array}{rcl} C & = & L_1^y - \dfrac{\lceil 2\langle T_1^y\rangle_2\rfloor_0}{2\beta} \\ \text{and} \quad D & = & \dfrac{\lceil 2\langle T_1^y\rangle_2\rfloor_0}{2\beta} - \arctan\left(d_1^y\beta^{-1}\right)\end{array}$$

The equation (13b) become

$$\overline{L}_1^y = C + D \quad (15)$$

By definition of C and T_1^y (see 7):

$$\begin{array}{rcl} |C| & \leq & \left|\dfrac{L_1^y\beta - \langle L_1^y\beta\rangle_2}{\beta}\right| + \left|\dfrac{2\langle L_1^y\beta\rangle_2 - \lceil 2\langle L_1^y\beta\rangle_2\rfloor_0}{2\beta}\right| \\ & \leq & \dfrac{1}{\beta^3} + \dfrac{1}{4\beta} \\ |C| & \leq & \dfrac{1}{2\beta}\end{array} \quad (16)$$

Furthermore, by definition of D and (12b):

$$|D| \leq \frac{1}{2\beta} \quad (17)$$

Then, from (15), (16) and (17) we have :

$$\begin{aligned} \left|\overline{L}_1^y\right| &\leq |C| + |D| \\ \left|\overline{L}_1^y\right| &\leq \frac{1}{\beta} \end{aligned}$$

The second step is similar. The equation (14) proves that the value d_1^x exists in the table. Let note:

$$\begin{aligned} C' &= \overline{L}_1^x - \lceil 2\langle \overline{T}_1^x \rangle_2 \rfloor_0 \\ \text{and} \quad D' &= \frac{\lceil 2\langle \overline{T}_1^x \rangle_2 \rfloor_0}{2\beta} - \ln\left(1 + d_1^x \beta^{-1}\right) \end{aligned}$$

By definition of $\overline{T}_1^x$ (see 7), $|C'| \leq \frac{1}{2\beta}$. Moreover, by (12a), $|D'| \leq \frac{1}{2\beta}$. So :

$$\begin{aligned} |L_2^x| &\leq \left|C'\right| + \left|D'\right| \quad \text{and} \quad & L_2^y &= \overline{L}_1^x \\ |L_2^x| &\leq \frac{1}{\beta} & |L_2^y| &\leq \frac{1}{\beta}. \end{aligned} \tag{18}$$

The two next half-iterations (step 2 and $2 + \frac{1}{2}$) are normal iteration using an extended set of digits. In fact,

$$|T_2^x|, |T_2^y| \leq \beta.$$

using the equations (9) on page 4, if $\beta \geq 10$:

$$\begin{aligned} |L^x| &\leq \frac{2}{\beta^2} \\ \text{and} \quad |L_y| &\leq \frac{2}{\beta^2}. \end{aligned}$$

So

$$\begin{aligned} L &\in P_E. \\ \text{and} \quad \exp(L_1) &= (1 + id_1^y)(1 + d_1^x)(1 + id_2^y)(1 + d_2^x)\exp(L) \end{aligned}$$

these iterations reduce the argument from $\mathcal{D}$ to the convergence domain of the algorithm.

4 The computation of complex logarithm : L-mode

The second mode of BKM can be done through a similar way. In this mode, regarding the value E_1^x and E_1^y, we have to construct d_n^x and d_n^y such that the sequence E_n^x converge to 1 and E_n^y converges to 0.

We use the following iteration:

$$\begin{aligned} \overline{E}_n &= E_n\left(1 + id_n^y \beta^{-n}\right) \\ E_{n+1} &= \overline{E}_n\left(1 + d_n^x \beta^{-n}\right) \end{aligned} \tag{19}$$

this means that, we first multiply E_n by a number of the form $\left(1 + id_n^y \beta^{-n}\right)$ in order to minimize the imaginary part of the product and next, we multiply $\overline{E}_n$ in order to minimize the real part of the product.

4.1 Digits selection

Our digits should be close to the exact values:

$$d_n^x = \left(\frac{1}{\overline{E}_n^x} - 1\right)\beta^n \quad \text{and} \quad d_n^y = -\frac{E_n^y}{E_n^x}\beta^n \tag{20}$$

Our goal is that E_n^x has to converge to 1 and E_n^y has to converge to 0. Near those values, we can do the following approximations:

$$\left(\frac{1}{\overline{E}_n^x}-1\right)\beta^n \simeq -\left(\overline{E}_n^x-1\right)\beta^n \quad \text{and} \quad \frac{E_n^y}{E_n^x}\beta^n \simeq E_n^y\beta^n \tag{21}$$

In order to use these approximations, we do the following definitions:

$$\begin{aligned} S_n &= (E_n-1)\beta^n \\ \overline{S}_n &= (\overline{E}_n-1)\beta^n \end{aligned}$$

Then, a good choice for d_n^x and d_n^y, can be :

$$d_n^y = -\lceil S_n^y \rfloor_0 \quad \text{and} \quad d_n^x = -\lceil \overline{S}_n^x \rfloor_0$$

Like the E-mode in section 3.1, it is better to use :

$$\begin{aligned} d_n^y &= -\lceil \langle S_n^y \rangle_2 \rfloor_0 \\ d_n^x &= -\lceil \langle \overline{S}_n^x \rangle_2 \rfloor_0 \end{aligned} \tag{22}$$

As in section 3, we will first prove that S_n is bounded using this digit selection if we start from a small domain and then propose a method to do the argument reduction.

4.2 Correctness of the approximation

Let note

$$a = \lceil |S_n^x| \rceil \quad \text{and} \quad b = \lceil |S_n^y| \rceil \tag{23}$$

We will studies the behaviour of one step of our algorithm. We have :

$$\begin{aligned} |d_n^x + S_n^x| &\leq \frac{1}{2}+\beta^{-2} \\ |d_n^y + S_n^y| &\leq \frac{1}{2}+\beta^{-2} \end{aligned}$$

By definition, $|d_n^y| \leq b$ and $d_n^y S_n^y \leq 0$, so :

$$\begin{aligned} a \leq\ & \overline{S}_n^x \leq a+b^2\beta^{-n} \\ -\frac{1}{2}-\beta^{-2}-ab\beta^{-n} \leq\ & \overline{S}_n^y \leq \frac{1}{2}+\beta^{-2}+ab\beta^{-n} \end{aligned} \tag{24}$$

By definition, a is an integer and $-a-b^2\beta^{-n}-\frac{1}{2} \leq d_n^x \leq a$, then :

$$|S_{n+1}^x| \leq \frac{\beta}{2}+\beta^{-1}+(a+b^2\beta^{-n}+\frac{1}{2})^2\beta^{-n+1} \tag{25a}$$

$$\begin{aligned} \left|S_{n+1}^y\right| &\leq \overline{S}_n^y(1+a\beta^{-n}) \\ &\leq \frac{\beta}{2}+\beta^{-1}+ab\beta^{-n+1}+\frac{a}{2}\beta^{-n+1}+a\beta^{-n-1}+a^2b\beta^{-2n+1} \end{aligned} \tag{25b}$$

If $\beta \geq 6$ and :

- if $n=2$, $a=2$ and $b=2$ then $|S_3^x|, |S_3^y| \leq \frac{\beta}{2}+1$,
- if $n \geq 3$, $a=b=\frac{\beta}{2}+1$, then $|S_{n+1}^x|, \left|S_{n+1}^y\right| \leq \frac{\beta}{2}+1$.

This means that if $n \geq 2$ the algorithm converges on the following domain:

$$P_L = \left[1-2\beta^{-2}, 1+2\beta^{-2}\right]+i\left[-2\beta^{-2}, 2\beta^{-2}\right]$$

4.3 Argument reduction

This domain is small because of the choice of d_n^x and d_n^y. This is based on the approximation (21):

$$\left(\frac{1}{\overline{E}_n^x}-1\right)\beta^n \simeq -\left(\overline{E}_n^x-1\right)\beta^n \quad \text{and} \quad \frac{E_n^y}{E_n^x}\beta^n \simeq E_n^y\beta^n$$

This approximation is only valid if E_n^x is close to 1, but it is not true for the first iteration. As in [ALB00b], we have to reduce the argument to a domain where this approximation is valid using an "improved" iteration (like in section3.3). For this, we will use a lookup table for choosing the digits d_1^x and an extended base 4β. We use a lookup table T indexed by S_1^x rounded 2 bits after the point.

More precisely, T is defined as follow:

$$\begin{array}{l} \forall n \in \{-2,-1,0,\cdots,4\beta\} \\ \qquad \begin{array}{rrcl} \text{If } n \geq 0 & T(n) & = & \left(\left\lceil \dfrac{4\beta}{1+\frac{n}{4\beta}} \right\rfloor_0 - 4\beta\right) \times \dfrac{1}{4}, \\ \text{else} & T(n) & = & 0. \end{array} \end{array}$$

This table verifies:

$$\begin{array}{llcl} \text{if} & x \in [-\frac{1}{2\beta}, 1] & & \\ \text{then} & \left|(1+x)(1+T(\lceil\langle 4x\beta\rangle_2\rfloor_0))\right| & \leq & \dfrac{1}{2\beta} \\ \text{and} & \left|(1+T(\lceil\langle 4x\beta\rangle_2\rfloor_0))\right| & \leq & 1 \end{array} \tag{26}$$

We start from a value

$$\begin{array}{rrcl} & z & = & 1+x+iy \\ \text{with} & x & \in & [0,1] \\ \text{and} & y & \in & [-\frac{1}{2},\frac{1}{2}]. \end{array}$$

Reduction to this domain is fairly straightforward. We do the second argument reduction with iterations presented in table 2:

Step $\frac{1}{2}$	$\left\{\begin{array}{rcl} d_0^x & = & T(\lceil\langle 4x\beta\rangle_2\rfloor_0) \\ E_1^x & = & (1+x)\times(1+d_0^x\beta^{-1}) \\ E_1^y & = & y\times(1+d_0^x\beta^{-1}) \end{array}\right.$
Step 1	$\left\{\begin{array}{rcl} d_1^y & = & \lceil\langle S_1^y\rangle_2\rfloor_0 \\ \overline{E}_1 & = & E_1\times(1+id_1^y\beta^{-1}) \end{array}\right.$
Step $1+\frac{1}{2}$	$\left\{\begin{array}{rcl} d_1^x & = & T(\left\lceil\left\langle 4(\overline{E}_1^y-1)\beta\right\rangle_2\right\rfloor_0) \\ E_2 & = & \overline{E}_1\times(1+d_1^x\beta^{-1}) \end{array}\right.$
Step 2	$\left\{\begin{array}{rcl} d_2^y & = & \lceil\langle S_2^y\rangle_2\rfloor_0 \\ \overline{E}_2 & = & E_2\times(1+id_2^y\beta^{-2}) \end{array}\right.$
Step $2+\frac{1}{2}$	$\left\{\begin{array}{rcl} d_2^x & = & \left\lceil\left\langle \overline{S}_2^y\right\rangle_2\right\rfloor_0 \\ E & = & \overline{E}_2\times(1+d_2^x\beta^{-2}) \end{array}\right.$

Table 2: The five step of L-mode argument reduction

Consequently:

- If

$$\begin{array}{rcccl} 0 & \leq & x & \leq & 1 \\ -\dfrac{1}{2} & \leq & y & \leq & \dfrac{1}{2}. \end{array}$$

- After step $\frac{1}{2}$, by (26)

$$\begin{aligned} |S_1^x| &\leq \frac{1}{2} \\ |S_1^y| &\leq \frac{\beta}{2}. \end{aligned}$$

- After step 1 by (24),

$$\begin{array}{rcccl} -\frac{1}{2} & \leq & \overline{S}_1^x & \leq & \frac{\beta}{4}+1 \\ -1 & \leq & \overline{S}_1^y & \leq & 1. \end{array}$$

- By (26), after step $1+\frac{1}{2}$:

$$\begin{aligned} |S_2^x| &\leq \frac{\beta}{2} \\ |S_2^y| &\leq \beta. \end{aligned}$$

- The to next half-iterations are regular BKM iterations using an extended set of digits. With equations (25a) and (25b), it is easy to show that:

$$E \in P_L.$$

We reduced the argument to the convergence domain of our algorithm and

$$\begin{aligned} \ln(z) = & -\ln(1+d_0^x\beta^{-1}) - \ln(1+id_1^y\beta^{-1}) - \ln(1+d_1^x\beta^{-1}) \\ & -\ln(1+id_2^y\beta^{-2}) - \ln(1+d_2^x\beta^{-2}) + \ln(E). \end{aligned}$$

Then it computes the complex logarithm on the domain

$$[1,2]+i[-\frac{1}{2},\frac{1}{2}].$$

5 Conclusion

We proposed in this paper an extension of the BKM algorithm to high radix which reduces the number of iterations. Compared to the Antello et all. CORDIC implementation, we use smaller set of digits and smaller lookup tables. By definition, BKM iterations are more complex than CORDIC's ones. But BKM provides the complex exponential and logarithm while CORDIC computes real valued functions. Furthermore CORDICusing high radix have the problem of the scaling factor which has to be computed and compensated. The extra operations are very close to our BKM iteration but are done in such a way that CORDIC only provide real value functions.

Compared to the Lewis's implementation, our algorithm uses similar split iterations, but the used numbers are larger and the digit choice involves larger lookup tables. Thus, the cost of each iteration is larger as well as in term of hardware resources and computing time.

This algorithm can be good substitute of the multipartite table methods for improved precision computation. Indeed, very large lookup tables are used for computing elementary functions compared to the wanted precision. BKM algorithm offers a framework with simple operations and relatively small tables to achieve the same work.

References

[ALB00a] E. Antelo, T. Lang, and J.D. Bruguera. Very-high cordic rotation based on selection by rounding. *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, 25(2):141–154, June 2000.

[ALB00b] E. Antelo, T. Lang, and J.D. Bruguera. Very-high radix circular cordic vectoring and unified rotation/vectoring. *IEEE Transactions on Computers*, 49(7):727–738, July 2000.

[BB84] J.M. Borwein and P.B. Borwein. The arithmetic-geometric mean and fast computation of elementary functions. *SIAM Review*, 26(3):351–366, July 1984.

[BI99] J.C. Bajard and L. Imbert. Evaluation of complex elementary functions : a new version of BKM. In F.T. Luk, editor, *Proceedings of SPIE, Advanced Signal Processing Algorithms, Architectures and Implementations IX*, volume 3807, pages 2 – 9, july 1999. Denver – USA.

[BKM94] J.C. Bajard, S. Kla, and J.M. Muller. BKM : A new complex algorithm for complex elementary functions. *IEEE Transactions on Computers*, 43(8):955–963, august 1994.

[Bre76] R.P. Brent. Fast multiple-precision evaluation of elementary functions. *Journal of the ACM*, 23(2):242–251, April 1976.

[dDT00] Florent de Dinechin and Arnaud Tisserand. Some improvement on multipartie table methods. Technical report, LIP, 2000.

[DM96] H. Dawid and H. Meyr. The differential cordic algorithm : Constant scale factor redundant implementation without correcting iterations. *IEEE Transactions on Computers*, 45(3):307–318, March 1996.

[DMF00] M. Daumas and C. Moreau-Finot. Exponential : implementation trade-offs for hundred bit precision. In J.C. Bajard, C. Frougny, P. Kornerup, and J.M. Muller, editors, *RNC4, Fourth "real Numbers and Computers"*, pages 61–74, April 2000.

[HT95] Hannes Hassler and Naofumi Takagi. Function evaluation by table look-up and addition. In S. Knowles and W.H. McAllister, editors, *Proccedings of the 12th IEEE Symposium on Computer Arithmetic*, pages 10–16. IEEE Computer Society Press, 1995.

[IMR00] L. Imbert, J.-M. Muller, and F. Rico. Radix-10 bkm algorithm for computing transcendentals on pocket computers. *Journal of VLSI SIGNAL PROCESSING SYSTEMS for Signal, Image and Video Thechnology*, 25(2):179–186, June 2000.

[Lew99] David Lewis. High-radix redundant cordic algorithms for complexe logarithmic number system arithmetic. In *Proceedings of the 14th symposium on Computer Arithmetic*, pages 194–203, 1999.

[Mul97] Jean-Michel Muller. *Elementary Functions, Algorithms and Implementation*. Birkhauser, Boston, 1997.

[SM95] Debjit Das Sarma and David W. Matula. Faithful bipartite rom reciprocal tables. In S. Knowles and W.H. McAllister, editors, *Proccedings of the 12th IEEE Symposium on Computer Arithmetic*, pages 10–16. IEEE Computer Society Press, 1995.

[Smi89] D.M. Smith. Efficient multiple-precision evaluation of elementary functions. *Mathematics of Computation*, 52(185):131–134, January 1989.

[Tan89] P. T. P. Tang. Table-driven implementation of the exponential function in ieee floating-point arithmetic. *ACM Transactions on Mathematical Software*, 15(2):144–157, June 1989.

[Vol59] J. Volder. The CORDIC computing technique. *IEEE Transactions on Computers*, 1959. Reprinted in E. E. Swartzlander, *Computer Arithmetic*, Vol. 1, IEEE Computer Society Press Tutorial, Los Alamitos, CA, 1990.

[Wal71] J.S. Walther. A unified algorithm for elementary functions. *Joint Computer Conference Proceedings*, 1971. Reprinted in E. E. Swartzlander, *Computer Arithmetic*, Vol. 1, IEEE Computer Society Press Tutorial, Los Alamitos, CA, 1990.