Fast evaluation of elementary functions with combined shift-and-add and polynomial methods

J-C. Bajard* M.D. Ercegovac† L. Imbert‡ F. Rico*

Abstract

This paper deals with the computation of elementary functions. We propose an efficient approach which combines shift-and-add and polynomial methods. It takes advantages of both, since some very simple computations (additions and shifts) reduce the complexity of the polynomial evaluation (i.e, reduce the number of multiplications). The proposed algorithm is more efficient in comparison with polynomial and shift-and-add methods, considered separately.

1 Introduction

In order to compute elementary functions, one can choose algorithms from different classes such as shift-and-add algorithms [11][12], polynomial approximations [6], and methods based on the arithmetic-geometric mean [5]. The last one is used when a very large precision is needed (e.g., several thousand digits).

Shift-and-add algorithms were developed for hardware implementation since each iteration uses simple operations (addition and multiplication by a power of the radix). Polynomial approximations methods have been the basis of software libraries, but the use of tables (memory) makes these algorithms attractive for hardware implementation [4]. In this paper, we propose a method which combines these two approaches to take advantage of both approaches.

In Section 1 we review the BKM algorithm [3] which belongs to the shift-and-add class (like CORDIC [11, 12] and additive/multiplicative normalization [7, 8]). This algorithm is very attractive since it is well adapted to redundant number system. In such systems, additions could be done in constant time. Polynomials methods are discussed in 2.2. For high precision these methods may require very high degree polynomials. With a use of tables to reduce the interval of computation, the degree of the polynomial can be reduced [9].

*Laboratoire d'Informatique, de Robotique et de Micro-électronique, Montpellier, France
†University of California at Los Angeles, Los Angeles, CA, USA
‡Laboratoire d'Informatique de Marseille, Université de Provence, Marseille, France

In Section 3, we introduce a new algorithm which combines both methods. We compute the first digits of the result using BKM, and then the remaining digits are computed with a polynomial approximation using an efficient evaluation scheme. The complexity of the algorithm depends on the degree of the polynomial, and therefore on the number of BKM iterations. Experimental results are discussed in Section 4 for exponentials and logarithms using a precision of 128 bits.

2 Previous methods

2.1 Shift-and-add algorithms

The BKM algorithm, belongs to the class of shift-and-add methods such as used by Briggs to build the first table of logarithms and CORDIC [11, 12]. BKM allows the computation of the complex logarithm and exponential functions, and thus it can be used to compute the real elementary functions such as *sin, cos, arctan, ln,* and *exp*. Unlike CORDIC, it has no scaling factor and, therefore, it allows the use of redundant number system without penalty. The BKM algorithm is based on the following iterations:

$$\left\{ \begin{array}{l} E_{n+1} = E_n\left(1 + d_n 2^{-n}\right) \\ L_{n+1} = L_n - \ln\left(1 + d_n 2^{-n}\right) \end{array} \right. \tag{1}$$

with a complex digit $d_n = d_n^r + i\, d_n^i$, and $d_n^r, d_n^i = -1, 0, 1$

It operates in two modes: E-mode for the exponential, and L-mode for the logarithm:

- **E-mode** : Find a sequence $\{d_n\}$ such that $L_n \rightarrow 0$. Then $E_n \rightarrow E_1 e^{L_1}$.
- **L-mode** : Find a sequence $\{d_n\}$ such that $E_n \rightarrow 1$. Then $L_n \rightarrow L_1 + \ln(E_1)$.

If E_1 and L_1 are correctly selected, n steps of iteration (1) produce n significant digits of the result. Convergence domains for these two modes are given by equations (2) and (3). Proofs are given in [3, 2].

$$L_1 \in \left\{ L = L^r + iL^i, -0.829 < L^r < 0.868, -0.749 < L^i < 0.749 \right\}, \tag{2}$$

$$E_1 \in \left\{ E = E^r + iE^i, 0.64 < E^r < 1.4, -\frac{2}{5} < E^i < \frac{2}{5} \right\}, \tag{3}$$

Argument reductions consist of a simple translation for the E-mode which gives a final product, and a shift and add operation for the L-mode which gives a final sum. Figure 1 illustrates a computation in the E-mode. Note a linear convergence of the L-part. The method that we propose takes advantage of this decrease.

k	E_k^r	E_k^i	L_k^r	L_k^i
1	1.0000000000	0.0000000000	0.5100000000	0.2900000000
2	1.5000000000	0.0000000000	0.1045348919	0.2900000000
3	1.8750000000	0.3750000000	-0.1382190160	0.0926044402
4	1.6406250000	0.3281250000	-0.0046876234	0.0926044402
5	1.6201171875	0.4306640625	-0.0066369436	0.0301856302
6	1.6066589355	0.4812927246	-0.0071249866	-0.0010542033
7	1.6066589355	0.4812927246	-0.0071249866	-0.0010542033
8	1.5941069126	0.4775326252	0.0007181909	-0.0010542033
9	1.5941069126	0.4775326252	0.0007181909	-0.0010542033
$\vdots$				
27	1.5957550946	0.4761937329	-0.0000000053	-0.0000000049
28	1.5957550827	0.4761937294	0.0000000022	-0.0000000049
29	1.5957550904	0.4761937252	-0.0000000015	-0.0000000011
30	1.5957550874	0.4761937243	0.0000000003	-0.0000000011
31	1.5957550894	0.4761937233	-0.0000000006	-0.0000000002
32	1.5957550886	0.4761937231	-0.0000000001	-0.0000000002
33	1.5957550884	0.4761937226	0.0000000001	0.0000000000
34	1.5957550886	0.4761937226	-0.0000000000	0.0000000000

Figure 1: Evaluation of $\exp(0.51 + i0.29)$ to 10 decimal digits with 34 BKM iterations.

2.2 Polynomial approximations

Another class of function evaluation methods is based on polynomial approximations. The well-known methods such as Taylor, Chebyshev and Minimax are used to approximate elementary functions on an interval by polynomials [10].

For instance, the degree n Taylor approximation of the exponential function is:

$$\exp(x) \simeq 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \cdots + \frac{x^n}{n!} \tag{4}$$

where the error for $|x| \leq 1$ is less than $\dfrac{1}{(n+1)!} \times \dfrac{x^{n+1}}{1-x}$.

For a given precision p, we evaluate the polynomial so that the maximum error is less than 2^{-p}. The degree of the polynomial depends on the approximation method (e.g. Taylor or minimax), and especially on the interval range. Using these methods in a rather large domain requires polynomials of large degree, and, therefore, has a long computational delay. A common way to reduce this delay is to transform the original interval to a smaller one.

3 Proposed algorithm: BKM + Polynomial

The main idea here is to reduce the interval in which a polynomial is evaluated. A direct table look-up requires the size of the table which is of the same order as the interval reduction. For example, to transform the interval $[0, 1]$ to an interval $[0, \beta^{-n}]$, a table of β^n entries is needed.

We propose to reduce the domain interval by performing several BKM iterations. In this case, the size of the required table is equivalent to the log(size) of the direct table look-up on the original interval.

We have shown previously that the proposed method is efficient for very large numbers (several thousand bits) in the E-mode [1]. Figure 2 illustrates the differences in computational delay between the three methods: (i) using *mupad* package, (ii) a table look-up (2^8 entries) with a polynomial method, and (iii) our method with the number of BKM iterations corresponding to (1/16) of the precision.

In Section 4 we discuss the performance of the proposed method in evaluating exponentials and logarithms to a 128-bit precision.

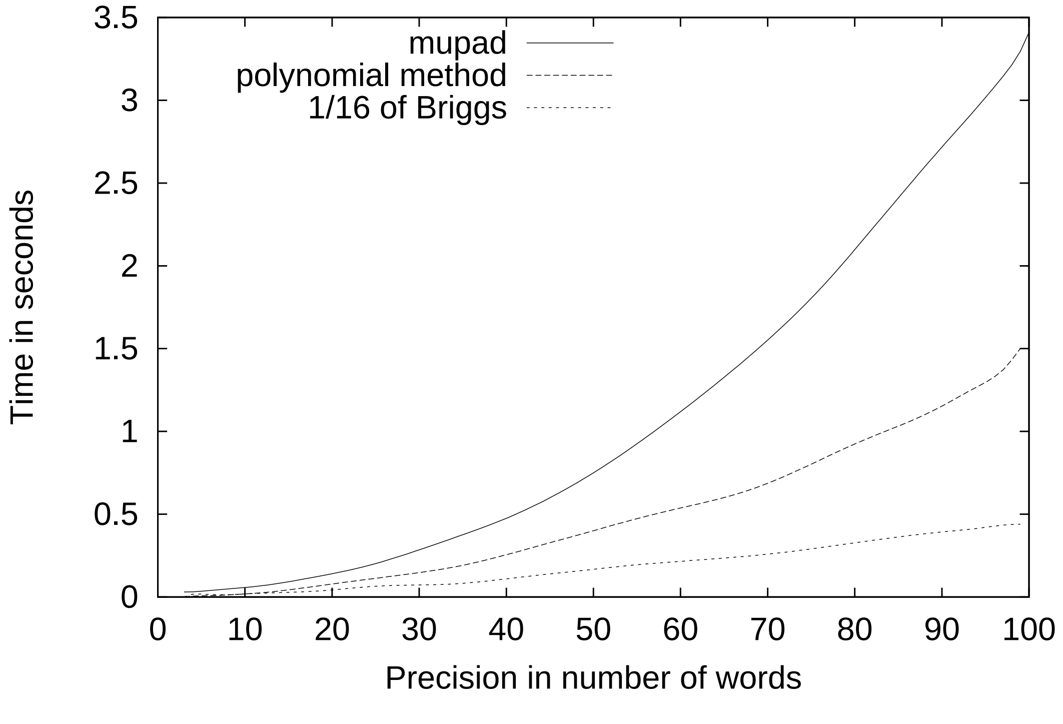

Figure 2: Computation of the real exponential

3.1 The exponential evaluation (using the E-mode)

If $L_1 = z$ and $E_1 = 1$, after n steps of BKM iterations, we obtain L_{n+1}, E_{n+1} and the sequence of digits $d_1, \ldots, d_n$ such that

$$\exp(z) \quad = \quad E_{n+1} \exp(L_{n+1}) \tag{5}$$

where

$$L_{n+1} \quad = \quad z - \sum_{k=1}^{n} \ln(1 + d_k 2^{-k}) \tag{6}$$

$$E_{n+1} \quad = \quad \prod_{k=1}^{n} (1 + d_k 2^{-k}) \tag{7}$$

We compute the real part $Re(\exp(z))$ and the imaginary part $Im(\exp(z))$ of $\exp(z)$ as follows:

$$Re(\exp(z)) \quad = \quad \exp(L_{n+1}^x) \times \left(E_{n+1}^x \times \cos(L_{n+1}^y) - E_{n+1}^y \times \sin(L_{n+1}^y)\right) \tag{8}$$

$$Im(\exp(z)) \quad = \quad \exp(L_{n+1}^y) \times \left(E_{n+1}^x \times \cos(L_{n+1}^y) + E_{n+1}^x \times \sin(L_{n+1}^y)\right) \tag{9}$$

where $\exp(L_{n+1}^x)$, $\sin(L_{n+1}^y)$ and $\cos(L_{n+1}^y)$ are evaluated with polynomial approximations. Note that

$$|L_n^x|, |L_n^y| \quad \leq \quad \frac{3}{2}\, 2^{-n}, \ \forall n$$

so that the exponential can be approximated by

$$T_n(x) \quad = \quad 1 + x + \frac{x^2}{2} + \cdots + \frac{x^n}{n!} \tag{10}$$

such that $|\exp(x) - T_n(x)| \leq \dfrac{x^{d+1}}{(d+1)!\,(1-x)}$

3.1.1 Error Analysis

In the E-mode of BKM E_1 is initialized to 1 and the $\log(p) - 1$ first iterations do not generate any error. Thus E_{n+1} is obtained with a relative error such that:

$$\tilde{E}_{n+1} = E_{n+1}(1 + c2^{-p+\delta_E}) \text{ with } \delta_E = \log(n - \log(p) + 1) \tag{11}$$

As for L_{n+1}, $L_1 = z$, and we obtain

$$\tilde{L}_{n+1} = L_{n+1} + c' 2^{-p+\delta_L} \quad \text{with } \delta_L = \log(n) \tag{12}$$

Note that $e^{z+\epsilon} = e^z e^\epsilon$ and if $|\epsilon| < 1$ then $|(e^\epsilon - 1)| < |\epsilon|(e-1)$.

In the polynomial part, since we use Horner scheme with an argument less than 0.5, we obtain an error independent of the degree:

$$\begin{aligned} \tilde{\exp}(\tilde{L}_{n+1}) &= \exp(\tilde{L}_{n+1})(1 + c''2^{-p}) \\ &= \exp(L_{n+1})(1 + c'(e-1)2^{-p+\delta_L})(1 + c''2^-p) \end{aligned} \quad (13)$$

Considering the last complex product, we obtain:

$$\begin{aligned} \tilde{\exp}(z) &= \tilde{\exp}(\tilde{L}_{n+1}) \times \tilde{E}_{n+1} \times (1 + c'''2^{-p}) \\ &= \exp(L_{n+1})E_{n+1}(1 + c2^{-p+\delta_E})(1 + c'(e-1)2^{-p+\delta_L})(1 + c''2^{-p}) \\ &= \exp(L_{n+1})E_{n+1}(1 + k2^{-p+\delta_L}) \end{aligned} \quad (14)$$

For example, when $n = 8$, $|k| < 2^4$. Compared to a direct polynomial method, the increase in the error is at most of $\delta_L = \log(n)$ bits.

3.1.2 Computation Time

To evaluate the polynomial for 128 bits of precision, the Horner scheme is sufficient. We have shown in [1] that the Smith scheme is more efficient for larger precision. Therefore, the time cost for evaluating a polynomial of degree $d(n)$ is $d(n)$ additions and $d(n)$ multiplications. We obtain a total time as:

$$(6n + 2d(n) + 2)A(p) + (2d(n) + 6)M(p), \quad (15)$$

where n is the number of BKM iterations, p is the number of bits used, $d(n)$ is the degree of the polynomial, and where $A(p)$ and $M(p)$ are the delay of a p-bits addition and a p-bits multiplication.

The interval reduction with the BKM iterations requires a table of $7n$ entries of p-bits numbers, compared a table of 2^{2n+1} entries of p-bits numbers needed to obtain the same reduction with a standard table lookup.

3.2 The logarithm evaluation (using the L-mode)

Let z be a complex number in the convergence domain of the L-mode of BKM. If we use as initial values $E_1 = z$, $L_1 = 0$, then after n iteration of BKM we get L_{n+1}, E_{n+1} and the sequence $d_1, \ldots d_n$ such that:

$$\ln(z) = L_{n+1} + \ln(E_1) \quad (16)$$

with

$$\begin{aligned} L_{n+1} &= -\sum_{k=1}^{n} \ln(1 + d_k 2^{-k}) \\ E_{n+1} &= E_1 \times \prod_{k=1}^{n}(1 + d_k 2^{-k}) \end{aligned} \quad (17)$$

Since $\ln(z)$ is a complex number, we have:

$$\begin{aligned} Re(\ln(z)) &= L^x_{n+1} + \frac{1}{2}\ln({E^x_{n+1}}^2 + {E^y_{n+1}}^2) \\ Im(\ln(z)) &= L^y_{n+1} + \arctan\left(\frac{E^y_{n+1}}{E^x_{n+1}}\right) \end{aligned} \tag{18}$$

We now use polynomial approximations to compute $\ln({E^x_{n+1}}^2+{E^y_{n+1}}^2)$ and $\arctan\left(\frac{E^y_{n+1}}{E^x_{n+1}}\right)$. Since polynomials are computed on reduced intervals, we have for $0.5 \leq E_1 \leq 1.5$, $n \geq 4$, $|E^x_n - 1| \leq 1.5 \times 2^{-n}$ and $|E^y_n| \leq 1.5 \times 2^{-n}$. Therefore

$$\begin{aligned} |{E^x_{n+1}}^2 + {E^y_{n+1}}^2 - 1| &\leq 3 \quad \times 2^{-n-1} \\ \frac{E^y_{n+1}}{E^x_{n+1}} &\leq \quad 2^{-n} \end{aligned}$$

The polynomials used for ln and arctan are:

$$LN(x) = \sum_{n=1}^{\infty}(-1)^{n-1}\frac{x^n}{n} \tag{19}$$

$$\begin{aligned} AT(x) &= \sum_{k=0}^{\infty}(-1)^k\frac{x^{2k+1}}{2k+1} \\ &= x \times \sum_{k=0}^{\infty}(-1)^k\frac{x^{2k}}{2k} \end{aligned} \tag{20}$$

3.2.1 Error Analysis

In the L-mode of BKM, L_1 is initialized to 0. During the first iterations, this value can stay null. Consequently L_{n+1} is obtained with a relative error such that

$$\tilde{L}_{n+1} = L_{n+1}(1 + c2^{-p+\delta_L}) \text{ with } \delta_L = \log(n) \tag{21}$$

E_1 is initialized to z, and as we compute the value $2^n(E_n - 1)$, we obtain

$$\tilde{E}_{n+1} - 1 = (E_{n+1} - 1)(1 + c'2^{-p+\delta_E}) \text{ with } \delta_E = \log(n) \tag{22}$$

Note that the function $z \mapsto \ln(1+z)$ is Lip 2 (satisfies the Lipschitz condition of order 2) if $1/2 \leq z \leq 2$. Then

$$\begin{aligned} \ln(\tilde{E}_{n+1}) &= \ln(1 + (E_{n+1} - 1)) + 2c'2^{-p+\delta_E}(E_{n+1} - 1) \\ &= \ln(E_{n+1})(1 - 4c'2^{-p+\delta_E}) \quad \text{as} \quad \frac{1}{2}|E_{n+1} - 1| \leq |\ln(E_{n+1})| \end{aligned} \tag{23}$$

Observing the polynomial part, we have

$$\begin{aligned}
\tilde{\ln}(z) &= \tilde{L}_{n+1} + \tilde{\ln}(\tilde{E}_{n+1}) && (24)\\
&= L_{n+1}(1 + c2^{-p+\delta_L}) + \ln(\tilde{E}_{n+1})(1 + c''2^{-p}) && (25)\\
&= L_{n+1}(1 + c2^{-p+\delta_L}) + \ln(E_{n+1})(1 - 4c'2^{-p+\delta_E})(1 + c''2^{-p})\\
&= (L_{n+1} + \ln(E_{n+1}))(1 + k2^{-p+\delta_L})\\
&= \ln(z)(1 + k2^{-p+\delta_L})
\end{aligned}$$

It can be assumed that $|k| < 2^5$.

3.2.2 Computation Time

Let $d(n)$ be the degree of the polynomial necessary to achieve the precision of p. This degree is the same for ln and arctan.

$$d(n) = \max\{i \in \mathbb{N} \; ; \; \left(1 - 2^{-n}\right) 2^{in}(i+1) < 2^p\} \qquad (26)$$

Table 2 gives the values of $d(n)$ for a precision of 128 bits.

The evaluation of the degree $d(n)$ Taylor polynomial for ln requires $d(n) - 1$ additions and $d(n)$ multiplications, and for arctan it requires $\left\lceil \frac{d(n)}{2} \right\rceil$ multiplications and $\left\lceil \frac{d(n)}{2} \right\rceil - 1$ additions. Hence, the total time is

$$\left(6n + d(n) + \left\lceil \frac{d(n)}{2} \right\rceil + 3\right) A(p) + \left(d(n) + \left\lceil \frac{d(n)}{2} \right\rceil + 2\right) M(p) + D(p), \qquad (27)$$

where $D(p)$ is the delay of a p-bits number division.

The tables needed for the BKM iterations are the same as for the exponential algorithm. In order to compare it with a standard lookup table reduction to the same interval, we use the formula

$$\begin{aligned}
z &= Z_t 2^{-n} + z_r 2^{-n} && (28)\\
ln(1+z) &= ln(1 + Z_t 2^{-n}) + ln\left(1 + \frac{z_r 2^{-n}}{1 + Z_t 2^{-n}}\right)
\end{aligned}$$

where Z_t a Gauss number, and $z_r \in [-1,1] + i[-1,1]$. Thus, $ln(1 + Z_t 2^{-n})$ and $\dfrac{1}{1 + Z_t 2^{-n}}$ are precomputed and we need a table of 2^{2n+2} real values of 128 bits.

4 Simulation results for 128-bits precision

The Figure 3 shows the computation time for exponential as a function of the number of BKM iterations. The minimum computation delay is obtained with 8 BKM iterations and a degree 12 polynomial.

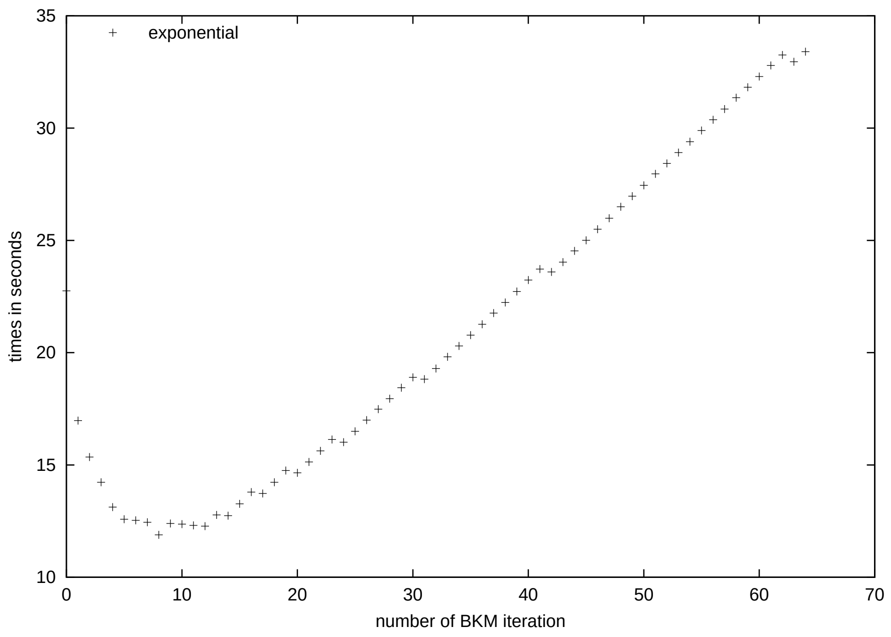

Figure 3: Delay for computing 100 000 exponential with 128 bits of precision.

If the number n of BKM iterations increases past 0, the cost of the BKM part increases linearly, but the degree of the polynomial (and the number of multiplication) decreases faster. As n increases, there are three distinct situations:

- The degree decreases by more than 2 ($n = 0$ to 4 and 7), and the delay of computation decrease sharply.
- The degree of polynomial decreases by 1 ($n = 5$, 6 and 9 to 11), then the cost increases by one BKM iteration and but decreases by two multiplications (one for the real part of the polynomial, one for the imaginary part). Thus, the global time decreases only slightly.
- The degree doesn't change (for example, for $n = 8$ to 9,or 31 to 41). In this case the increase in the delay is due to the extra BKM iterations.

The figure 4 shows the computation time for the complex logarithm using n BKM iterations. The behavior in this case is similar to the exponential. In both cases, the minimum delay is obtain for $n = 8$. In Figure 5 and 6 we plotted on a log scale the computation delay as a function of the table (memory) size. We observe that for the exponential and logarithm, we need a 2^{11} entries table to be faster than our algorithm using 8 step of BKM and only 57 precomputed values. Therefore, our algorithm is also efficient in reducing the size of the table.

n	$degree$	n	$degree$	n	$degree$	n	$degree$	n	$degree$	n	$degree$
0	34	6	15	12	9	18	7	$\cdots$	$\cdots$	$\cdots$	$\cdots$
1	28	7	14	13	9	19	7	30	5	62	3
2	24	8	12	14	8	20	6	31	4	63	2
3	21	9	12	15	8	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
4	18	10	11	16	8	23	6	41	4	127	2
5	16	11	10	17	7	24	5	42	3	128	1

Table 1: Degree of the polynomial for the exponential function

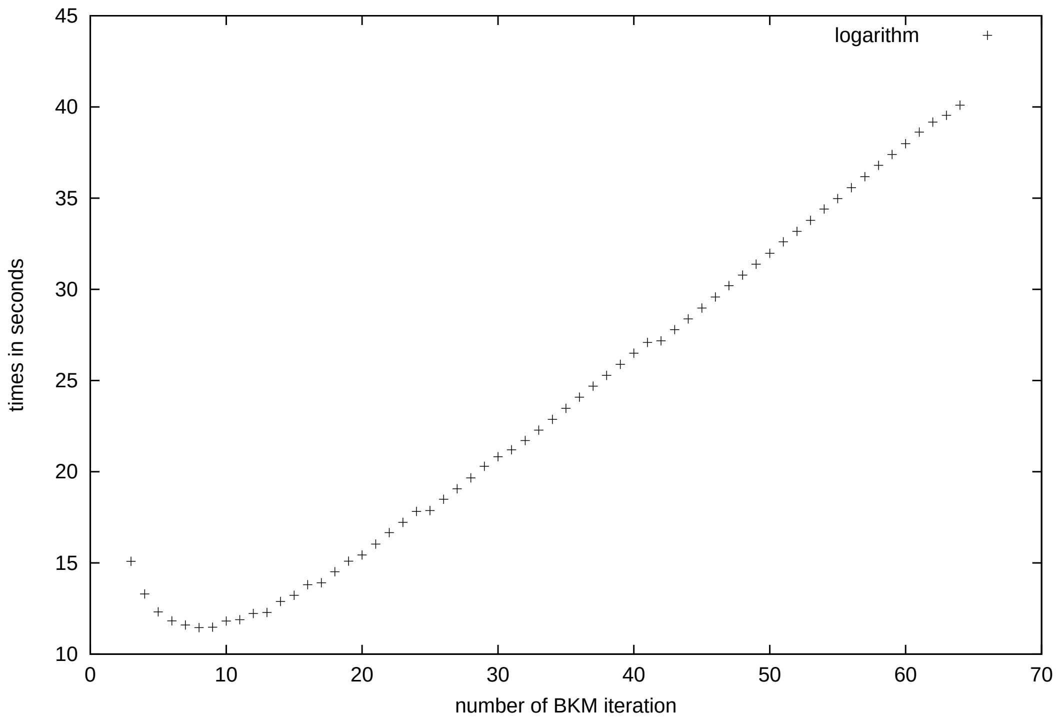

Figure 4: Delay for computing 100 000 logarithm with 128 bits of precision.

n	$degree$	n	$degree$	n	$degree$	n	$degree$	n	$degree$	n	$degree$
1	123	7	18	13	10	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
2	62	8	16	14	9	20	7	31	5	63	3
3	41	9	14	15	9	21	6	32	4	64	2
4	31	10	13	16	8	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
5	25	11	12	17	8	25	6	42	4	127	2
6	21	12	11	18	7	26	5	43	3	128	1

Table 2: Degree of the polynomial for the logarithm function

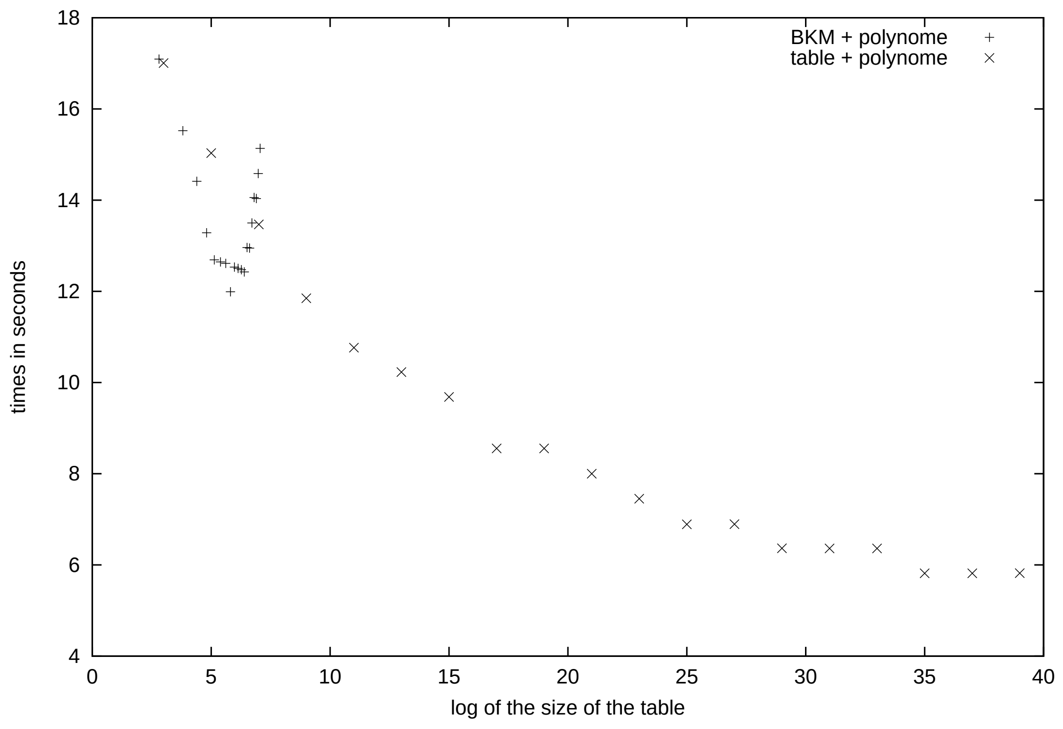

Figure 5: The delay vs. the size of memory in evaluating exponential.

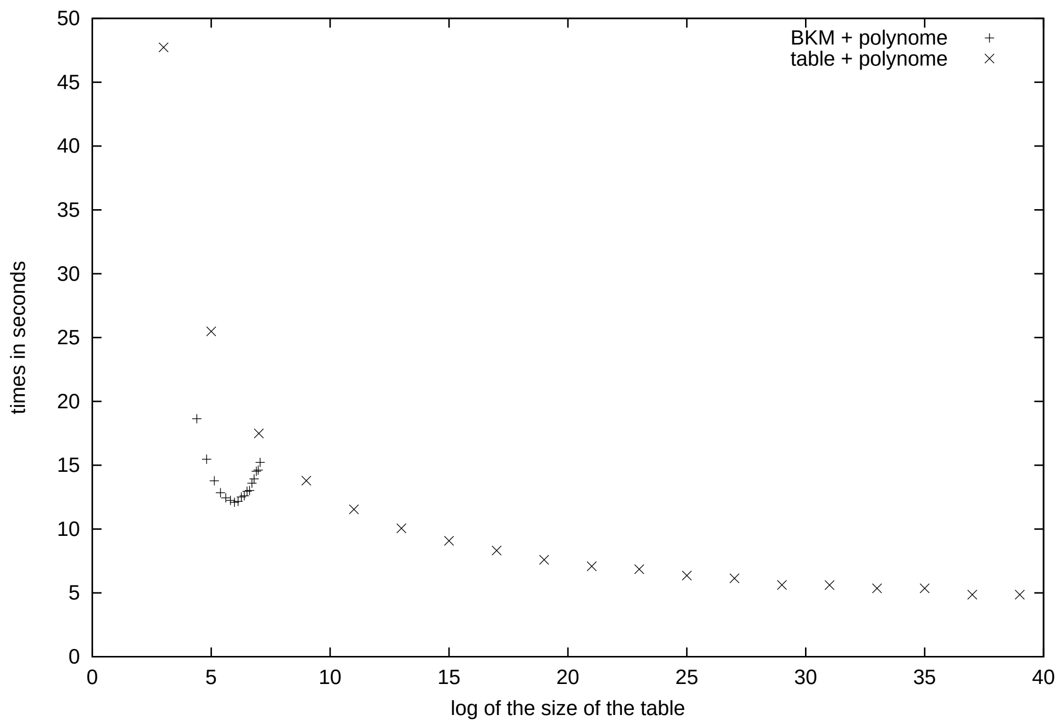

Figure 6: The delay vs. the size of memory in evaluating logarithm.

5 Summary

We have presented a new approach for the evaluation of elementary functions which combines shift-and-add (BKM) and polynomial approximations methods. Compared with classical table-based methods and polynomial approximations, our approach uses very small tables and requires the evaluation of low degree polynomials.

References

[1] J-C. Bajard, L. Imbert, and F. Rico. Evaluation de l'exponentielle du sinus et du cosinus à base d'additions de décalages et de polynômes. In *SympA'5* 5^{ieme} *symposium en architectures nouvelles de machines*, pages 19 – 26, Irisa, Rennes - FRANCE, juin 1999.

[2] 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.

[3] 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.

[4] K. Braune. Standard functions for real and complex point and interval arguments with dynamic accuracy. *Computing*, Suppl 6:159–184, 1988.

[5] R.P. Brent. Fast multiple precision evaluation of elementary functions. *Journal of the ACM*, 23:242–251, 1976.

[6] W.J. Cody. A survey of practical rational and polynomial approximation of functions. *SIAM Review*, 12(3):400–423, july 1970.

[7] B. de Lugish. *A class of algorithms for automatic evaluation of certain elementary functions in a binary computer.* PhD thesis, University of Illinois, Urbana, june 1970.

[8] M.D. Ercegovac. Radix 16 evaluation of certain elementary functions. *IEEE Transactions on Computers*, C-22(6), june 1973. Reprinted in E. E. Swartzlander, *Computer Arithmetic*, Vol. 1, IEEE Computer Society Press Tutorial, Los Alamitos, CA, 1990.

[9] S. Gal. Computing elementary functions: A new approach for achieving high accuracy and good performance. In *Accurate Scientific Computations. Lecture Notes in Computer Science*, volume 235, pages 1–16. Springer-Verlag, Berlin, 1986.

[10] J.M. Muller. *Elementary functions,* Algorithms and Implementation. Birkhäuser, 1997.

[11] 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.

[12] 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.