

# Radix-10 BKM Algorithm for Computing Transcendentals on Pocket Computers

Laurent Imbert, Jean-Michel Muller, Fabien Rico

► **To cite this version:**

Laurent Imbert, Jean-Michel Muller, Fabien Rico. Radix-10 BKM Algorithm for Computing Transcendentals on Pocket Computers. [Research Report] RR-3754, LIP RR-1999-43, INRIA, LIP. 1999. inria-00072908

**HAL Id: inria-00072908**

**<https://hal.inria.fr/inria-00072908>**

Submitted on 24 May 2006

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

***Radix-10 BKM Algorithm for Computing  
Transcendentals on Pocket Computers***

Laurent Imbert , Jean-Michel Muller , Fabien Rico

**No 3754**

Septembre 1999

————— THÈME 2 —————

A large blue rectangle occupies the lower half of the page. Overlaid on it is the text 'Rapport de recherche' in a white serif font. A large, light grey 'R' is positioned to the left of the word 'Rapport'. A horizontal grey brushstroke is located below the word 'recherche'.

**R**apport  
de recherche





## Radix-10 BKM Algorithm for Computing Transcendentals on Pocket Computers

Laurent Imbert <sup>\*</sup>, Jean-Michel Muller <sup>†</sup>, Fabien Rico<sup>‡</sup>

Thème 2 — Génie logiciel  
et calcul symbolique  
Projet Arénaire

Rapport de recherche n° 3754 — Septembre 1999 — 12 pages

**Abstract:** We present a radix-10 variant of the BKM algorithm. It is a shift-and-add, CORDIC-like algorithm that allows fast computation of complex exponentials and logarithms. This radix-10 version is suitable for implementation in a pocket computer.

**Key-words:** Elementary functions, CORDIC algorithms, Computer Arithmetic, radix-10 arithmetic

*(Résumé : tsvp)*

<sup>\*</sup> Laboratoire d'Informatique de Marseille

<sup>†</sup> CNRS-LIP, projet CNRS-ENSL-INRIA Arénaire

<sup>‡</sup> Laboratoire d'Informatique de Marseille

## Un algorithme BKM de base 10 pour calculer des fonctions transcendantes sur des calculatrices de base 10

**Résumé :** Nous proposons une variante de l'algorithme BKM adaptée au calcul en base 10. C'est un algorithme à additions et décalages, qui permet d'évaluer rapidement des exponentielles et logarithmes complexes. Cette version adaptée à la base 10 est destinée à l'implantation sur des calculatrices de poche.

**Mots-clé :** Fonctions élémentaires, algorithmes CORDIC, Arithmétique des ordinateurs, arithmétique base 10

## 1 Introduction

Our goal is to design fast shift-and-add algorithms for implementing elementary functions of a complex variable on pocket calculators. As a matter of fact, the most well known shift-and-add algorithm, namely the CORDIC algorithm [?, ?], has been the first algorithm used for evaluating transcendentals on a pocket computer. Recent surveys can be found in [?, ?]. Several authors recently suggested redundant [?] versions of CORDIC, to get faster iterations [?, ?, ?, ?].

In this paper, we aim at designing a radix-10, redundant version of the BKM algorithm. The BKM algorithm was introduced by Bajard, Kla and Muller in 1993 [?]. It is a CORDIC-like algorithm that allows the computation of complex logarithms and exponentials (and therefore of most real elementary functions), without scale factor (even when using redundant arithmetic). Bajard and Imbert have recently suggested some variants of this algorithm [?].

Whereas the original algorithm was suited for radix 2 arithmetic, if we want to design very efficient implementations for pocket computers, we need a radix-10 variant of the algorithm, since most, if not all, pocket computers use a radix-10 arithmetic.

Our algorithm requires a special first step whose purpose is to reduce the initial argument to some value that is small enough, so that the subsequent iterations will converge. After that, the BKM iterations are applied:

$$\begin{cases} E_{n+1} = E_n(1 + d_n 10^{-n}) \\ L_{n+1} = L_n - \ln(1 + d_n 10^{-n}) \end{cases}$$

where  $d_n$  is such that a multiplication by  $d_n$  is straightforwardly performed in radix-10 arithmetic.

### 1.1 Notations

- $\forall x \in \mathbb{C}$ , we note  $\langle x \rangle_p$ , the values of the real and imaginary parts of  $x$  **rounded to the nearest number** with  $p$  fractional digits. For example, we note  $\langle x \rangle_0$  the point in  $\mathbb{Z} + i\mathbb{Z}$  closest to  $x$ .
- $\forall x \in \mathbb{C}$ , we note  $\lfloor x \rfloor_p$ , the values of the real and imaginary parts of  $x$  **truncated down** with  $p$  fractional digits.

## 1.2 The BKM algorithm

From the initial values  $L_1$  and  $E_1$ , we perform in parallel the following 2 iterations:

$$\begin{cases} L_{n+1} = L_n - \ln(1 + d_n 10^{-n}) \\ E_{n+1} = E_n(1 + d_n 10^{-n}) \end{cases}$$

with:

$$\begin{aligned} d_n &= d_n^x + i d_n^y \\ -6 &\leq d_n^x, d_n^y \leq 6 \end{aligned}$$

The value  $d_n$  depends on which function we wish to calculate. The BKM algorithm has 2 modes:

- **E-mode:** if we find a sequence  $d_n$  such that  $L_n$  goes to 0, then we will obtain  $E_n \rightarrow E_1 e^{L_1}$ .
- **L-mode:** if we find a sequence  $d_n$  such that  $E_n$  goes to 1, then we will obtain  $L_n \rightarrow L_1 + \ln(E_1)$ .

## 2 The complex exponential function (E-mode)

In this section, we will focus on the problem of finding a sequence  $d_n$  such that  $L_n$  goes to 0.

If we consider the iteration  $L_{n+1} = L_n - \ln(1 + d_n 10^{-n})$ , the choice that immediately springs in mind for  $d_n$  (but of course it is not implementable) is  $d_n = \langle (e^{L_n} - 1) 10^n \rangle_0$ . Let us define  $T_n = 10^n L_n$ . We will first assume that  $L_n$  is very small compared to 1. We will later see how to reduce the general case to this one. Our previous ‘‘ideal’’ choice for  $d_n$  is equivalent to choosing  $d_n = \langle T_n \rangle_0$ .

In the following, we suppose that  $L_n \ll 1$ . We describe one iteration of the algorithm and prove the convergence for  $n \geq 3$  in the domain:  $L_3 \in [-0.0065, 0.0065] + i[-0.0065, 0.0065]$ .

### 2.1 One step of the algorithm

We give an algorithm for which we will be able to show by induction that the real and imaginary parts of  $T_n$  have absolute value less than 6 for any  $n$ .

We suppose that, after  $n$  iterations:

$$-6 < T_n^x, T_n^y < 6 \tag{1}$$

We will prove that:  $-6 < T_{n+1}^x, T_{n+1}^y < 6$ .

The iteration  $L_{n+1} = L_n - \ln(1 + d_n 10^{-n})$  becomes:

$$\begin{aligned} L_{n+1}^x &= L_n^x - \frac{1}{2} \ln(1 + 2d_n^x 10^{-n} + (d_n^{x2} + d_n^{y2}) 10^{-2n}) \\ L_{n+1}^y &= L_n^y - d_n^y \arctan\left(\frac{10^{-n}}{1 + d_n^x 10^{-n}}\right) \end{aligned}$$

If we consider the Taylor series for  $\ln(1+x)$ , and  $\arctan(x)$ , we obtain:

$$\begin{aligned} \frac{1}{2} \ln(1 + 2d_n^x 10^{-n} + (d_n^{x2} + d_n^{y2}) 10^{-2n}) &\simeq d_n^x 10^{-n} \\ d_n^y \arctan\left(\frac{10^{-n}}{1 + d_n^x 10^{-n}}\right) &\simeq d_n^y 10^{-n} \end{aligned}$$

So we need  $d_n^x$  to be very close to  $T_n^x$  and  $d_n^y$  close to  $T_n^y$ . We can choose  $d_n^x = \langle [T_n^x]_2 \rangle_0$  and  $d_n^y = \langle [T_n^y]_2 \rangle_0$ . That means that we choose as  $d_n^x$  (resp.  $d_n^y$ ) the digit closest to  $T_n^x$  (resp.  $T_n^y$ ) considering the first 3 digits of  $T_n^x$  (resp.  $T_n^y$ ) only. Note that  $-6 \leq d_n^x, d_n^y \leq 6$ .

Then

$$|[T_n^x]_2 - T_n^x|, |[T_n^y]_2 - T_n^y| \leq \frac{1}{100} \quad \text{and}$$

$$|d_n^x - [T_n^x]_2|, |d_n^y - [T_n^y]_2| \leq \frac{5}{10}.$$

So

$$|d_n^x - T_n^x|, |d_n^y - T_n^y| \leq \frac{51}{100}$$

Let us define:

$$\begin{aligned} A_n &= \frac{1}{2} \ln(1 + 2d_n^x 10^{-n} + (d_n^{x2} + d_n^{y2}) 10^{-2n}) \\ B_n &= d_n^y \arctan\left(\frac{10^{-n}}{1 + d_n^x 10^{-n}}\right) \end{aligned}$$

then

$$\begin{aligned} |T_{n+1}^x| &\leq 5, 1 + (d_n^x 10^{-n} - A_n) 10^{n+1} \\ |T_{n+1}^y| &\leq 5, 1 + (d_n^y 10^{-n} - B_n) 10^{n+1} \end{aligned}$$

So, we need:

$$|d_n^x 10^{-n} - A_n| 10^{n+1} \leq 0.9 \quad (2)$$

$$|d_n^y 10^{-n} - B_n| 10^{n+1} \leq 0.9 \quad (3)$$



In order to prove (2), let us define:  $x = 2d_n^x 10^{-n} + (d_n^{x^2} + d_n^{y^2}) 10^{-2n}$ .  
 If  $n \geq 3$ , we have:  $|x| \leq \frac{1}{75}$ .  
 We define:  $\alpha = 75 \ln\left(\frac{75}{76}\right)$ , and  $\beta = 75 \ln\left(\frac{75}{74}\right)$ .  
 We know that (if  $x \geq 0$ ):

$$\alpha x \leq \ln(1+x) \leq x$$

Thus:

$$d_n^x (\alpha - 1) 10^{-n} + \frac{\alpha}{2} (d_n^{x^2} + d_n^{y^2}) 10^{-2n} \leq A_n - d_n^x 10^{-n} \leq \frac{1}{2} (d_n^{x^2} + d_n^{y^2}) 10^{-2n}$$

If  $n \geq 3$  and  $|d_n^x|, |d_n^y| \leq 6$ , as  $\alpha \geq 0.993$  we have

$$|A_n - d_n^x 10^{-n}| 10^{n+1} \leq 0.9$$

similarly if  $x \leq 0$ ,

$$\beta x \leq \ln(1+x) \leq x$$

and as  $\beta \leq 1.007$

$$|A_n - d_n^x 10^{-n}| 10^{n+1} \leq 0.9$$

for (3), we can say that (if  $x \geq 0$ ):

$$x - \frac{x^3}{3} \leq \arctan(x) \leq x$$

and then, if  $n \geq 3$ :

$$d_n^y \frac{10^{-n}}{1+d_n^x 10^{-n}} - \frac{d_n^y}{3} \left( \frac{10^{-n}}{1+d_n^x 10^{-n}} \right)^3 \leq B_n \leq d_n^y \frac{10^{-n}}{1+d_n^x 10^{-n}}$$

As  $|d_n^x|, |d_n^y| \leq 6$ ,

$$|B_n - d_n^y 10^{-n}| \leq 0.037 \times 10^{-n}.$$

If  $x \leq 0$  we have

$$x \leq \arctan(x) \leq x - \frac{x^3}{3}$$

and the same result.

So the domain of convergence of the algorithm is

$$P = [-0.0065, 0.0065] + i[-0.0065, 0.0065]$$

we can extend this domain to

$$P = [-0.025, 0.035] + i[-0.035, 0.035]$$

since if we reduce the number of values for  $d_2^x$ , and  $d_2^y$ , we can obtain  $L_2 \in P$ . We have plotted the surfaces corresponding to  $z = |d_2^x 10^{-2} - A_2| 10^3$ ,  $z = |d_2^y 10^{-2} - B_2| 10^3$ , and the plane  $z = 0.9$ , for  $-2 \leq d_2^x \leq 3$ , and  $-3 \leq d_2^y \leq 3$ .

We can see in figure 1 that:

$$\begin{aligned} |d_2^x 10^{-2} - A_2| 10^3 &\leq 0.9 \\ |d_2^y 10^{-2} - B_2| 10^3 &\leq 0.9 \end{aligned}$$

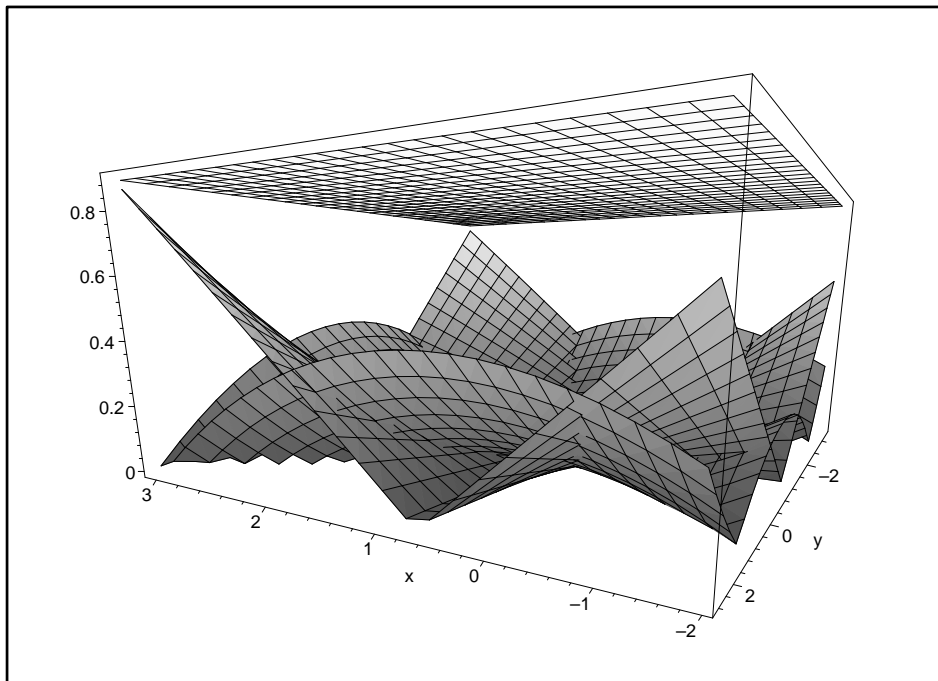


Figure 1: Error on  $P$

Unfortunately, this allow the computation of  $\exp(x)$  in a tiny domain only. In the next section, we will give a method to reduce  $x$  to the convergence domain.

## 2.2 Range reduction for the exponential

We assume that we wish to evaluate  $\exp(x)$ , where  $x = a + ib$ , and  $-\frac{1}{2} \leq a, b \leq \frac{1}{2}$ . Reduction to this domain is fairly straightforward.

Define  $b^*$  as the number obtained by truncating  $b$  to two fractional digits. From  $b^*$ , we look up in a table the number of the form  $\beta_k = \log(1 + ik_b 10^{-2})$ , with  $-55 \leq k_b \leq 55$  that is closest to  $b^*$ . The number  $y = x - \log(1 + ik_b 10^{-2})$  has an imaginary part of absolute value less than  $\frac{1}{100} + \frac{1}{2} \max |\Im(\beta_{k+1} - \beta_k)| = 0.01 + 0.0049998333433326191032$ .

The real part of  $y$  is between  $-\frac{1}{2} - 0.133$  and  $\frac{1}{2}$ . Define  $a^*$  as  $Re(y)$  truncated to two fractional digits. From  $a^*$  we look up in a table the number of the form  $\alpha_k = \log(1 + ik_a 10^{-2})$ , with  $-47 \leq k_a \leq 65$  that is closest to  $a^*$ . The number  $z = y - \log(1 + ik_a 10^{-2})$  has the same imaginary part as  $y$ , and its real part has an absolute value less than  $\frac{1}{100} + \frac{1}{2} \max |\alpha_{k+1} - \alpha_k| = 0.01 + 0.00934606650607627473$ .

We will choose  $z$  as the reduced argument. Its real part has an absolute value less than 0.02, and its imaginary part has an absolute value less than 0.015.

## 3 The complex logarithm function (L-mode)

In this section, we will focus on the problem of finding a sequence  $d_n$  such that  $E_n$  goes to 1.

If we note:  $S_n = 10^n(E_n - 1)$ , the choice that immediately springs in mind for  $d_n$  (but of course it is not implementable) is  $d_n = -\lfloor \frac{S_n}{E_n} \rfloor$ .<sup>1</sup> It is clear, that if  $E_n \simeq 1$ , this is equivalent to choosing  $d_n = -\lfloor S_n \rfloor$ .

In the following, we suppose  $E_n \simeq 1$ . We describe one iteration of the algorithm and prove the convergence for  $n \geq 2$ . In section 3.2, we will explain the reduction step that gives  $E_n \simeq 1$ .

### 3.1 One step of the algorithm

We suppose that, after  $n$  iterations:

$$-6 < S_n^x, S_n^y < 6 \quad (4)$$

The iteration  $E_{n+1} = E_n(1 + d_n 10^{-n})$  becomes:

$$\begin{aligned} S_{n+1}^x &= 10 (S_n^x + d_n^x + (d_n^x S_n^x - d_n^y S_n^y) 10^{-n}) \\ S_{n+1}^y &= 10 (S_n^y + d_n^y + (d_n^x S_n^y + d_n^y S_n^x) 10^{-n}) \end{aligned}$$

<sup>1</sup>Actually,  $E_n(1 - \frac{S_n}{E_n} 10^{-n}) = 1$

so we need  $d_n^x$  to be very close to  $-S_n^x$  and  $d_n^y$  to be very close to  $-S_n^y$ . We can choose  $d_n^x = -\langle \lfloor S_n^x \rfloor_2 \rangle_0$  and  $d_n^y = -\langle \lfloor S_n^y \rfloor_2 \rangle_0$ . That means that we choose as  $d_n^x$  (resp.  $d_n^y$ ) the digit closest to  $-S_n^x$  (resp.  $-S_n^y$ ) considering only the first 3 digits of  $S_n^x$  (resp.  $S_n^y$ ).

Then

$$|\lfloor S_n^x \rfloor_2 - S_n^x|, |\lfloor S_n^y \rfloor_2 - S_n^y| \leq \frac{1}{100} \quad \text{and}$$

$$|d_n^x + \lfloor S_n^x \rfloor_2|, |d_n^y + \lfloor S_n^y \rfloor_2| \leq \frac{5}{10}.$$

So

$$|d_n^x + S_n^x|, |d_n^y + S_n^y| \leq \frac{51}{100}$$

and then

$$\begin{aligned} |S_{n+1}^x| &\leq 10 \left( \frac{51}{100} + (|d_n^x S_n^x| + |d_n^y S_n^y|) 10^{-n} \right) \\ &\leq 5, 1 + (|d_n^x S_n^x| + |d_n^y S_n^y|) 10^{-n+1} \\ |S_{n+1}^y| &\leq 5, 1 + (|d_n^x S_n^y| + |d_n^y S_n^x|) 10^{-n+1} \end{aligned}$$

If  $n \geq 3$ , as  $|S_n^x|, |S_n^y| < 6$  and  $|d_n^x|, |d_n^y| < 6$ ,

$$\begin{aligned} 10^{-n+1} (|d_n^x S_n^x| + |d_n^y S_n^y|) &\leq 0.72 < 0.9 \\ 10^{-n+1} (|d_n^x S_n^y| + |d_n^y S_n^x|) &\leq 0.72 < 0.9 \end{aligned}$$

If  $n = 2$ , and if  $|S_2^x|, |S_2^y| < 2$  then,  $|d_2^x|, |d_2^y| < 2$ , and

$$\begin{aligned} 10^{-1} (|d_2^x S_2^x| + |d_2^y S_2^y|) &\leq 0.8 < 0.9 \\ 10^{-1} (|d_2^x S_2^y| + |d_2^y S_2^x|) &\leq 0.8 < 0.9 \end{aligned}$$

We obtain:

- if  $S_2 \in [-2, 2] + i[-2, 2]$  then  $\|S_3\|_\infty < 6$ ,
- and for  $n \geq 3$ , by induction  $\|S_n\|_\infty < 6$ .

As a result, we have an algorithm which converge to  $\ln(x)$  for

$$x \in P = [0.98, 1.02] + i[-0.02, 0.02]$$

Unfortunately, this allows the computation of  $\ln(x)$  in a tiny domain only. In order to allow the computation for more values, we need to reduce the argument.

### 3.2 Range reduction for the logarithm

Using a method presented in [?], we can reduce the argument to

$$S = \{x + iy; x > 0 \text{ and } -\frac{2}{5}x \leq y \leq \frac{2}{5}x\}$$

From  $S$ , it is easy to reduce to  $T$  (one shift and one multiplication by 2)

$$T = \{x + iy; 0.98 \leq x \leq 2 \text{ and } -\frac{2}{5}x \leq y \leq \frac{2}{5}x\}$$

So, our goal is to find a reduction from  $T$  to  $P$ .

In the following, we will explain this step in greater detail.

- start with  $E_1 \in T$
- choose  $d = \langle \lfloor E_1 \rfloor_3 \rangle_1$
- read in a table  $z = \left\langle \frac{1}{1 + \frac{d}{10}} \right\rangle_2$
- compute  $\hat{z} = E_1 \times z$
- choose  $d' = \langle \lfloor \hat{z} \rfloor_4 \rangle_2$
- read in a table  $z' = \left\langle \frac{1}{1 + \frac{d'}{100}} \right\rangle_2$
- compute  $E_2 = \hat{z} \times z'$  ( $E_2 \in P$ )

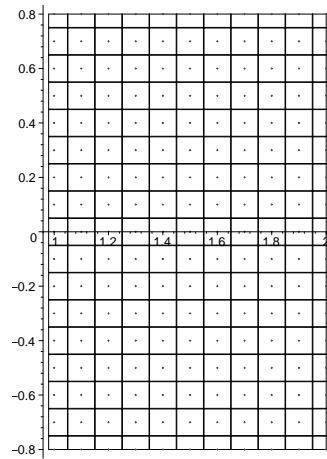
#### Proof of the reduction

First, we define  $R$ , a rectangle such that  $T \subset R$ . We split  $R$  into 187 squares depending of the different choices for  $d$  (see figure 2(a)). Then, we have computed the image of  $R$  by the first step of the argument reduction (that is to say the image of each square by a similitude). We can see on figure 2(b) that this image is enclosed into the square:

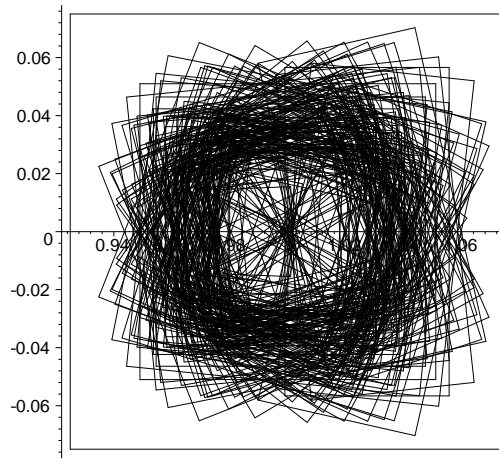
$$\hat{R} = [0.925, 1.075] + i[-0.075, 0.075].$$

The second step is similar. We split  $\hat{R}$  into 225 squares, and compute by the same way, the image of  $\hat{R}$  (see figure 2(c), and 2(d)).

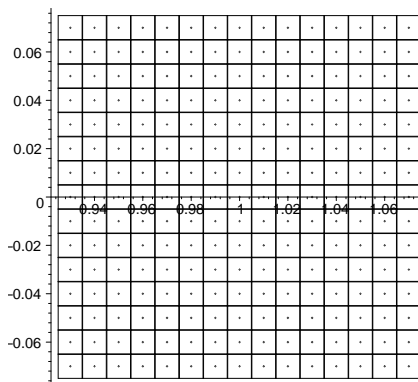
Consequently, all the computed points ( $E_2$ ) are enclosed into  $P$ .



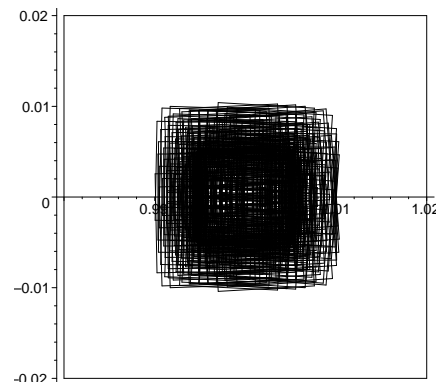
(a) cutting out of  $R$



(b) image of  $R(z)$



(c) cutting out of  $\hat{R}$



(d) image of  $\hat{R}(E_2)$

Figure 2: argument reduction for the logarithm function

## 4 Conclusion

We have presented a radix-10 variant of the BKM algorithm. Extensions of this work to other radices (not too large, though) should be straightforward. A radix-10 BKM algorithm could be implemented into pocket calculators, to get fast complex elementary functions.

## References

- [1] A. Avizienis. Signed-Digit Number Representation for Fast Parallel Arithmetic. *IRE Transaction on electronic computers*, 10:389–400, 1961.
- [2] J.C. Bajard and L. Imbert. Evaluation of complex elementary functions : new version of BKM. In *SPIE International Symposium on Optical Science, Advanced Signal Processing Algorithms, Architectures and Implementations*, 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] 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.
- [5] J. Duprat and J. M. Muller. The CORDIC algorithm: New results for fast VLSI implementation. *IEEE Transactions on Computers*, 42(2):168–178, February 1993.
- [6] M. D. Ercegovac and T. Lang. Redundant and on-line CORDIC: Application to matrix triangularization and SVD. *IEEE Transactions on Computers*, 39(6):725–740, June 1990.
- [7] G. J. Henkstra. *CORDIC for High-Performance Numerical Computation*. PhD thesis, TU Delft, 1998.
- [8] J.M. Muller. *Elementary Functions*. Birkhäuser, 1997.
- [9] S. Yajima T. Asada, N. Takagi. Redundant Cordic Methods with a Constant Scale Factor. *IEEE Transactions on Computers*, 40(9), 1991.
- [10] J. Volder. The cordic computing technique. *IEEE Transactions on Computers*, 1959.
- [11] J.S. Walther. A unified algorithm for elementary functions. *Joint Computer Conference Proceedings*, 1971.



---

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,  
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY  
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex  
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN  
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex  
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

---

Éditeur  
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)  
<http://www.inria.fr>  
ISSN 0249-6399