An Efficient Method for Evaluating Polynomial and Rational Function Approximations

Nicolas Brisebarre[1], Sylvain Chevillard[1], Miloš D. Ercegovac[2], Jean-Michel Muller[1] and Serge Torres[1]
[1] LIP - Arénaire (CNRS - ENS Lyon - INRIA - UCBL), École Normale Supérieure de Lyon,
46, allée d'Italie, F-69364 Lyon Cedex 07, France, firstname.lastname@ens-lyon.fr
[2] Computer Science Department, 4732 Boelter Hall, University of California at Los Angeles,
Los Angeles, CA 90095, USA, milos@cs.ucla.edu

Abstract

In this paper we extend the domain of applicability of the E-method [7, 8], as a hardware-oriented method for evaluating elementary functions using polynomial and rational function approximations. The polynomials and rational functions are computed by solving a system of linear equations using digit-serial iterations on simple and highly regular hardware. For convergence, these systems must be diagonally dominant. The E-method offers an efficient way for the fixed-point evaluation of polynomials and rational functions if their coefficients conform to the diagonal dominance condition. Until now, there was no systematic approach to obtain good approximations to f over an interval $[a, b]$ by rational functions satisfying the constraints required by the E-method. In this paper, we present such an approach which is based on linear programming and lattice basis reduction. We also discuss a design and performance characteristics of a corresponding implementation.

1. Introduction

Rational approximations are rather seldom used for approximating functions in the common math libraries, because division is much slower than multiplication. There are no hardware-oriented methods for rational functions, with the notable exceptions of the E-method [7, 8] or approaches which use conventional arithmetic operations and table-lookups [13]. The E-method, although intended for hardware implementation, can be viewed as a general approach for software implementations. However, the method, as reviewed shortly, is not directly applicable to any rational function.

The evaluation method (E-method), introduced in [7, 8], maps a rational function (ratio of two polynomials) or a polynomial to a system of linear equations which is then solved using digit-by-digit approach on a simple and highly regular hardware implementation. The method requires that the resulting linear system is diagonally dominant which poses a problem in using arbitrary rational functions. The goal of the present work is to remove this limitation and make the E-method an attractive general approach for function evaluation.

Let

$$R_{\mu,\nu}(x) = \frac{P_\mu(x)}{Q\nu(x)} = \frac{p_\mu x^\mu + p_{\mu-1}x^{\mu-1} + \cdots + p_0}{q_\nu x^\nu + q_{\nu-1}x^{\nu-1} + \cdots + q_1 x + 1}$$

where the p_is and q_is are real numbers, and define $n = \max\{\mu, \nu\}$, $p_j = 0$ for $\mu + 1 \le j \le n$, and $q_j = 0$ for $\nu + 1 \le j \le n$. According to the E-method $R_{\mu,\nu}(x)$ is mapped to a linear system $L : \mathbf{A} \times \mathbf{y} = \mathbf{b}$:

$$\begin{bmatrix} 1 & -x & 0 & & \cdots & & 0 \\ q_1 & 1 & -x & 0 & \cdots & & 0 \\ q_2 & 0 & 1 & -x & & \cdots & 0 \\ & & \ddots & \ddots & \ddots & & \vdots \\ & & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & \ddots & \ddots & 0 \\ & & & & & 1 & -x \\ q_n & & & \cdots & & 0 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ \vdots \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ \vdots \\ \vdots \\ p_{n-1} \\ p_n \end{bmatrix}$$

so that $y_0 = R_{\mu,\nu}(x)$.

In the radix-2 case this linear system is solved digit-by-digit, computing all solution components in parallel, by using the following residual recurrence where $\mathbf{A}$ is the matrix of the system L and $\mathbf{w}$'s are residuals:

$$\mathbf{w}^{(j)} = 2 \times \left[\mathbf{w}^{(j-1)} - \mathbf{A}\mathbf{d}^{(j-1)}\right]$$

for $j = 1, \ldots, m$, where m is the precision of the result. The individual components y_i's of the solution $\mathbf{y}$ are computed using the following digit-recurrences:

$$w_i^{(j)} = 2 \times \left[w_i^{(j-1)} - q_i d_0^{(j-1)} - d_i^{(j-1)} + d_{i+1}^{(j-1)} x\right], \quad (1)$$

1-4244-1898-5/08/$20.00 ©2008 IEEE

$$w_0^{(j)} = 2 \times \left[w_0^{(j-1)} - d_0^{(j-1)} + d_1^{(j-1)} x \right]$$

and

$$w_n^{(j)} = 2 \times \left[w_n^{(j-1)} - d_n^{(j-1)} - q_n d_0^{(j-1)} \right]$$

with $\mathbf{w}^{(0)} = [p_0, p_1, \ldots, p_n]^t$ and the values $d_i^{(j)} \in \{-1, 0, 1\}$. Let $D_i^{(j)} = d_i^{(0)}.d_i^{(1)} d_i^{(2)} \ldots d_i^{(j)}$ represent in radix-2 signed-digit system the leftmost j digits of the solution component y_i. If the residuals $|w_i^{(j)}|$ are bounded, then $D_i^{(j)} \to y_i$ as $j \to +\infty$. Digits $d_i^{(j)}$ are selected so that the residuals remain bounded. The digit selection can be performed by rounding of the residuals [8, 9]:

$$\begin{aligned} d_i^{(j)} &= S(w_i^{(j)}) \\ &= \begin{cases} \text{sign } w_i^{(j)} \times \left\lfloor \left| w_i^{(j)} \right| + 1/2 \right\rfloor, & \text{if } \left| w_i^{(j)} \right| \le 1, \\ \text{sign } w_i^{(j)} \times \left\lfloor \left| w_i^{(j)} \right| \right\rfloor, & \text{otherwise.} \end{cases} \end{aligned}$$

Moreover, this selection can be performed using a low-precision estimate $\widehat{w}_i^{(j)}$ of $w_i^{(j)}$, obtained by truncating or rounding redundant (i.e., carry-save or borrow-save) $w_i^{(j)}$.

Define the following bounds :

$$\begin{cases} \forall i, |p_i| \le \xi, \\ \forall i, |x| + |q_i| \le \alpha, \\ |w_i^{(j)} - \widehat{w}_i^{(j)}| \le \frac{\Delta}{2}. \end{cases} \tag{2}$$

where Δ is the size of the overlap between the selection intervals where the choice of the digit is made.

The E-method requires that the above defined bounds ξ, α, and Δ satisfy

$$\begin{cases} \xi = \frac{1}{2}(1 + \Delta), \\ 0 < \Delta < 1, \\ \alpha \le \frac{1}{4}(1 - \Delta). \end{cases} \tag{3}$$

For instance, if $\Delta = \frac{1}{2}$, one can evaluate $R(x)$ for $|x| \le \frac{1}{16}$, $\max |p_i| \le \frac{3}{4}$ and $\max |q_i| \le \frac{1}{16}$. Those bounds are restrictive, but in practice they present no difficulties for polynomials since there are scaling techniques to achieve the required bounds. However, this is not the case for rational functions in general. In certain cases a scaling can be found but not for *all* rational functions. To remove this limitation we propose to derive rational function approximations, called *simple E-fractions*, which are products of a power of 2 by a fraction that satisfies the bounds (2) and (3). In [4], we give a more general definition of *E-fractions*, where some additional scaling is allowed.

In the following sections, we show how a general function can be approximated by a simple E-fraction. We develop in Section 2 an algorithmic approach based on linear programming and lattice basis reduction that provides us with a simple E-fraction. In Section 3, we apply the method of Section 2 to two examples. Then, we explain in Section 4 the design and the implementation of the method.

2. Effective computation of simple E-fractions

In this section we present our approach. We combine a linear programming process that first gives us a simple E-fraction with real coefficients with a lattice basis reduction based method that makes it possible to compute, from the real-coefficient simple E-fraction given by the first step, a simple E-fraction with fixed-point or floating-point coefficients.

2.1. Linear programming step

Let f be a continuous function defined on $[a, b]$. Let r, $s \in \mathbb{N}$ be given. The aim of this subsection is to compute a (very) good rational fraction approximant F, with respect to the infinite norm defined by $||g|| = \sup_{x \in [a,\, b]} |g(x)|$, of f such that the degree of the numerator of F is less or equal to r, the degree of the denominator of F is less or equal to s, and the real coefficients of F (or F divided by some fixed power of 2) satisfy the constraints imposed by the E-method.

The first thing to do is to apply the classical Remez algorithm [6] to get $R^\star$, the best possible rational approximant to f. Then we determine the least integer j such that the coefficients of the numerator of $R^\star$ divided by 2^j fulfill the first condition of (2). It gives us a decomposition $R^\star(x) = 2^j R_1(x)$. If the computed coefficients of R_1 fulfill the constraints (2), then we directly go to subsection 2.2. If not, we develop the following process.

We want to find $\epsilon > 0$, as small as possible, such that F is a simple E-fraction that satisfies $|f(x) - F(x)| \le \epsilon$.

If F is given as $F(x) = \frac{\sum_{j=0}^{r} p_j^\star x^j}{1 + \sum_{j=1}^{s} q_j^\star x^{j+1}}$, the constraints of the method imply that its denominator is positive on $[a, b]$. That is to say that we want to find ϵ, as small as possible, solution of the following problem: for all $x \in [a, b]$,

$$\begin{cases} \sum_{j=0}^{r} p_j^\star x^j - \sum_{j=1}^{s} (f(x) - \epsilon) q_j^\star x^j \ge f(x) - \epsilon, \\ -\sum_{j=0}^{r} p_j^\star x^j + \sum_{j=1}^{s} (f(x) + \epsilon) q_j^\star x^j \ge -f(x) - \epsilon, \\ q_j^\star \ge -\alpha + \min(|a|, |b|), -q_j^\star \ge -\alpha + \min(|a|, |b|), \forall j. \end{cases} \tag{4}$$

We get rid of the conditions $|p_j^\star| \le \xi$ by dividing F by a suitable power of 2.

We use linear programming to find an approximation of the optimal ϵ. We first plug numerous points of $[a, b]$ into (4), this gives us a polyhedron $\mathfrak{P}(\epsilon)$, and we search for a "fairly" small ϵ using the following iterative process.

We will use two variables denoted ϵ_l and ϵ_u. Let ϵ_R denote the error given by Remez' algorithm (that is to say the best possible), we start the process with $\epsilon_l = \epsilon_R$ and $\epsilon_u = 2\epsilon_R$. In the sequel of this subsection, the value ϵ_l is indeed a lower bound for the optimal value of ϵ whereas ϵ_u is only a heuristic candidate for being an upper bound for this optimal ϵ. We choose a parameter η that will be used to monitor the accuracy of the error we output. We have to be able to check if for a given ϵ, the corresponding polyhedron $\mathfrak{P}(\epsilon)$ is empty or not, and if not, to return a point of the polyhedron. We use a simplex-based algorithm, implemented in Maple, to that end.

The iteration is the following:

- while the relative error $|\epsilon_l/\epsilon_u - 1| \leq \eta$, do
  1. as long as the polyhedron $\mathfrak{P}(\epsilon_u)$ is empty, update the values: $\epsilon_l = \epsilon_u$ and $\epsilon_u = 2\epsilon_u$;
  2. (dichotomy) set $\epsilon_u = (\epsilon_l + \epsilon_u)/2$. Repeat this step until the polyhedron $\mathfrak{P}(\epsilon_u)$ is empty;
  3. set $temp = \epsilon_l$, $\epsilon_l = \epsilon_u$ and $\epsilon_u = \epsilon_u + (\epsilon_u - temp)/2$. Go to step 2;
- return a point of the polyhedron.

The algorithm returns the real coefficients of a good rational approximation to f, which moreover is a simple E-fraction.

2.2. Lattice basis reduction step

In order to implement practically the E-method, we need to get rational fractions with fixed-point or floating-point coefficients. In this subsection, we explain how to get a good rational fraction with such coefficients from a rational fraction with real coefficients. More precisely, given a simple E-fraction of the form

$$F(x) = \frac{\sum_{j=0}^{r} p_j^\star x^j}{1 + \sum_{j=1}^{s} q_j^\star x^{j+1}}$$

defined on a closed interval $[a,\ b]$, our procedure will return a simple E-fraction of the form

$$\widehat{F}(x) = \frac{\sum_{j=0}^{r} p_j x^j}{1 + \sum_{j=1}^{s} q_j x^{j+1}}$$

where p_j and q_j are fixed-point or floating-point numbers.

The method is heuristic: we do not provide a theoretical proof that the returned fraction will have a good quality of approximation. We do not either have a proof that the method will return a simple E-fraction. However, as will be seen in Section 3, the method gives good results on practical examples and if F is a simple E-fraction, $\widehat{F}$ will generally be one as well.

2.2.1 Notations concerning fixed-point and floating-point numbers

We assume the reader is familiar with fixed-point and floating-point arithmetic (see [9] for details). We just want here to define our notations. Fixed-point numbers and floating-point numbers will be of the form $m2^e$ where $m \in \mathbb{Z}$, with the following differences:

- for fixed-point numbers, e is implicit (decided at programming time);
- for floating-point numbers, e is explicit (i.e., stored). A floating-point number is of precision t if $2^{t-1} \leq m < 2^t - 1$ (e.g. the binary representation of m requires t bits exactly).

A different format can be used for each coefficient of the desired fraction. For instance, the user may choose that:

- p_0 is a fixed-point number with $e = -10$,
- p_1 is a floating-point number with precision $t = 25$,
- q_0 is a floating-point number with precision $t = 17$, etc.

We assume that we are given the desired format for each coefficient.

2.2.2 Formalization

We will keep in mind that, in general, F is an approximation to a continuous function f on the interval $[a,\ b]$. We want to get a rational fraction $\widehat{F}$ of the same form as F but with floating-point (or fixed-point) coefficients and providing a good approximation as well.

A coefficient that should be a fixed-point number leads to a single unknown m. One that should be a floating-point number with precision t leads to two unknowns m and e.

In [3], a method was presented that lets one find good polynomial approximants with floating-point coefficients. The procedure that we describe herein is just the adaptation of this method to the case of rational fractions.

A heuristic trick is given in [3] to find an appropriate value for e in the floating-point case. Basically, it consists in assuming that p_j will have the same order of magnitude as $p_j^\star$ (the same holds for q_j and $q_j^\star$). Once e is correctly guessed, the condition $2^{t-1} \leq |m| < 2^t$ may be forgotten and the problem is reduced to a fixed-point one. See [3] for practical details.

At this point, there just remain $r + s + 1$ unknown integers: one for each coefficient.

2.2.3 Determining the mantissas

From now on, we will suppose that the exponents have been correctly guessed and we focus on solving the following problem (P): given parameters e_j and f_j, find integers m_j and n_j such that the following fraction is a good approximation of F (and f):

$$\frac{\sum_{j=0}^{r} m_j 2^{e_j} x^j}{1+\sum_{j=1}^{s} n_j 2^{f_j} x^{j+1}}.$$

To this end, we choose $r+s+1$ points $x_1 < \cdots < x_{r+s+1}$ in the interval $[a,\, b]$ and we express the fact that we want our rational fraction to be close to the ideal one F at these points:

$$\forall i = 1 \ldots r+s+1, \quad \frac{\sum_{j=0}^{r} m_j 2^{e_j} x_i^j}{1+\sum_{j=1}^{s} n_j 2^{f_j} x_i^{j+1}} \simeq F(x_i).$$

This can be rewritten under the following form: $\forall i = 1 \ldots r+s+1$,

$$\sum_{j=0}^{r} m_j 2^{e_j} x_i^j - F(x_i)\left(\sum_{j=1}^{s} n_j 2^{f_j} x_i^{j+1}\right) \simeq F(x_i).$$

Thus, we have a linear system of $r+s+1$ unknowns and $r+s+1$ equations. This system could be exactly solved if m_i and n_i were real values and it would lead to F itself. But we want the solution of the system to be composed by integers. A method could be to replace each coefficient of the exact solution by the closest integer. This is what the authors of [3] called *the naive method* since it could lead to a significant loss of accuracy. The method that we propose will often give much better results.

Indeed, the system can be rewritten with vectors, emphasizing its linearity: we try to solve the following problem where m_i and n_i are integers:

$$\sum_{j=0}^{r} m_j \begin{pmatrix} 2^{e_j} x_1^j \\ \vdots \\ 2^{e_j} x_{r+s+1}^j \end{pmatrix} - \sum_{j=1}^{s} n_j \begin{pmatrix} 2^{e_j} x_1^{j+1} F(x_1) \\ \vdots \\ 2^{e_j} x_{r+s+1}^{j+1} F(x_{r+s+1}) \end{pmatrix} \simeq \begin{pmatrix} F(x_1) \\ \vdots \\ F(x_{r+s+1}) \end{pmatrix}.$$

In this equation, all the vectors are parameters and the unknowns are the m_j and n_j. As expressed in [3] it is an instance of a problem known as *the closest vector problem in a lattice*. Indeed, a lattice is the mathematical structure formed by all linear combinations with integer coefficients of a family of linearly independent vectors. This is exactly the structure that appears in our formalization. We are looking for a point in the lattice that is as close as possible to the vector $(F(x_1), \ldots, F(x_{r+s+1}))^t$.

To formally measure how close two vectors in $\mathbb{R}^{r+s+2}$ are, we need to set a norm. The choice of it is not crucial: the discretization already introduced a method error with respect to our original problem; the choice of a norm in $\mathbb{R}^{s+t+2}$ just introduces another one. Reasonable choices are the infinite norm $\|(v_1, \ldots, v_{r+s+2})\|_\infty = \max_i |v_i|$ and the euclidean norm $\|(v_1, \ldots, v_{r+s+2})\|_2 = (\sum_i v_i^2)^{1/2}$. In the sequel we will consider the euclidean norm but we could use the infinite norm as well.

To find an approximation of the closest vector, we use the same method as the authors of [3]: we use the LLL algorithm [12] and Babai's algorithm [2] that give us satisfying integer coefficients. The interested reader will find a detailed description of this method in [3]. These algorithms require the euclidean norm. Other (more costly) algorithms exist for the infinite norm.

There remains one problem: how to choose the points? The authors of [3] suggest to choose the points that would lead to the best possible solution if the interpolation system were solved exactly (e.g. with real coefficients). In our case, every choice of points would lead to F itself. However, since we want our fraction $\widehat{F}$ to be close to f as well, it could be a good idea to include the points x such that $F(x) = f(x)$: this way, f and F are simultaneously interpolated. Chebyshev's points, a classical option, would also be satisfying in general.

3. Some examples

In this section, we will look for fractions with fixed-point coefficients on 24 bits: these coefficients will be of the form $i/2^{24}$, $-2^{24} \leq i \leq 2^{24}$.

Example 1. The function we consider is $\sinh$ over $[0, 1/8]$, approximated by a fraction with degree-3 numerator and degree-4 denominator. This example is given in the appendix of [8]. The parameters are $\xi = 3/4$ and $\alpha = 1/8$.

The error provided by Remez' algorithm is $6.3524...10^{-18}$ and the fraction obtained is a simple E-fraction. Hence, we directly apply the approach described in subsection 2.2. We obtain the following simple E-fraction $F(x) = ((2034138/2^{24})x^3 - (99734/2^{24})x^2 + x)/((-13065/2^{24})x^4 + (16638/2^{24})x^3 - (762065/2^{24})x^2 - (99734/2^{24})x + 1)$ that provides an error of approximation equal to $9.0150...10^{-14}$, i.e., an accuracy of 43 bits.

Example 2. Now, we deal with one example, inspired by [11], where the rational approximant returned by Remez algorithm does not satisfy diagonal dominance.

The function we consider is $\arctan$. We want to approximate it over $[0, \tan(\pi/32)]$ with a fraction of the form $xR_{3,4}(x^2)$ where $R_{3,4}$ is a rational fraction with degree-3 numerator and degree-4 denominator. The parameters are $\xi = 5/8$ and $\alpha = 3/8$.

The error provided by Remez' algorithm is $2.6322...10^{-27}$. Our linear programming-based process returns a real-coefficient fraction that leads to an error of $1.20690...10^{-13}$. The lattice-based approach gives a solution $R_{3,4}(x) = 2((-1499350/2^{24})x^3 + (1673762/2^{24})x^2 - (2796207/2^{24})x + (8388608/2^{24}))/((-1443542/2^{24})x^4 - (617822/2^{24})x^3 - (7827/2^{24})x^2 - (9/2^{24})x + 1)$ that provides an error of $5.5442...10^{-12}$.

There is an important loss of accuracy in that example and that kind of problems shall be addressed in our future works. And yet, the accuracy given by this approximant is 37 bits, which is sufficient in many applications.

4. Design and implementation of the method

The E-method is suitable for hardware implementation because the primitive modules are digit-vector multiplexers, redundant adders of $[3:2]$ or $[4:2]$ type, and registers. The overall structure consists of $n+1$ elementary units (EU), interconnected digit-serially. As mentioned earlier, the method computes one digit of each element of the solution vector per iteration in the MSDF (Most Significant Digit First) manner. The EUs operate concurrently. The time to obtain the solution to t digits of precision is about t cycles (iterations).

A general scheme for evaluation of rational functions is shown in Figure 1, illustrated for $n = 4$. A bit-parallel bus transmits x, p and q values in a broadcast mode. Note that the initialization cycles could be shorter than the iteration cycles.

The digit-serial outputs of EU can be converted into digit-parallel form using an on-the-fly converter OFC [9].

The EUi (radix 2) implements the residual recurrence (1) and the corresponding digit selection function

$$d_i^{(j)} = S(\widehat{w_i^{(j)}})$$

Its organization is shown in Figure 2.

It consists of a [4:2] adder, two multiple generators which are implemented with multiplexers, and four registers. The output digit selection S is a simple gate network. There are also four registers, storing q_i and x, and a sum-carry representation of the residual. Initally, $ws_i^{(0)} \leftarrow p_i$ and $wc_i^{(0)} \leftarrow 0$.

The cycle time of the elementary unit EU, in terms of a

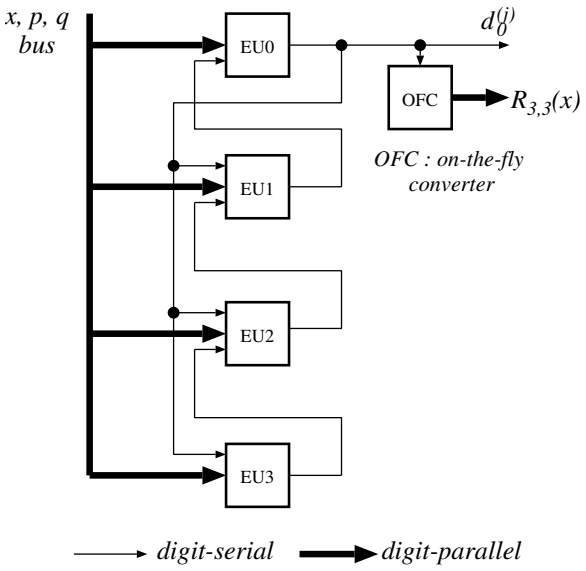

Figure 1. Scheme for evaluating rational function $R_{3,3}(x) = P_3(x)/Q_3(x)$ ($n = 4$).

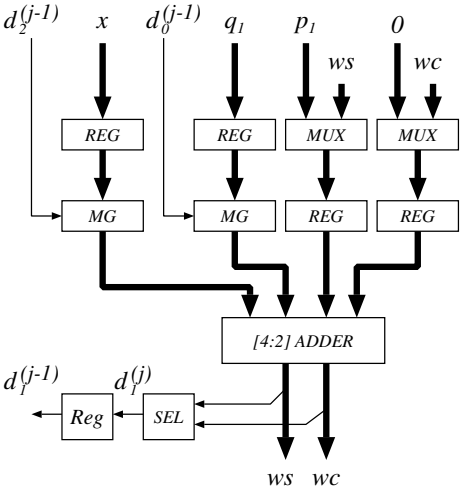

Figure 2. EU block diagram and organization ($i = 1$).

full adder (complex gate) delay t_{FA}, is estimated as follows

$$\begin{aligned} T_{EU} &= t_{BUFF} + t_{MG} + t_S + t_{[4:2]} + t_{REG} \\ &\approx (0.4 + 0.3 + 1 + 1.3 + 0.9)t_{FA} = 3.9t_{FA} \end{aligned}$$

The cost in terms of area of a full adder A_{FA} is estimated as

$$\begin{aligned} A_{EU}(t) &= A_S + 2A_{BUFF} + (t+2)[2A_{MG} \\ &+ A_{MUX} + A_{[4:2]} + 4A_{REG} + A_{OFC}] \\ &\approx [5 + 2 \times 0.4 + (t+2)(3 \times 0.45 \\ &+ 2.3 + 4 \times 0.6 + 2.1)]A_{FA} \\ &\approx 16 + 8tA_{FA} \end{aligned}$$

The cost is estimated as area occupied by modules using a full-adder A_{FA} area as the unit. The areas of primitive modules are [10]: Register $A_{REG} = 0.6A_{FA}$; buffer $A_{BUFF} = 0.4A_{FA}$; MUX $A_{MUX} = 0.45A_{FA}$; multiple generator MG $A_{MG} = 0.45A_{FA}$; [4:2] adder $A_{[4:2]} = 2.3A_{FA}$; S $A_S = 5A_{FA}$, and on-the-fly converters $A_{OFC} = 2A_{MUX} + 2A_{REG} = 2.1A_{FA}$.

5 Summary and future work

We have extended the domain of applicability of a hardware-oriented method for evaluating elementary functions using polynomial and rational function approximations. The polynomials and rational functions are computed by solving a system of linear equations using digit-serial iterations on simple and highly regular hardware. For convergence, these systems must be diagonally dominant and, consequently, the coefficients of rational functions must satisfy certain conditions which limit direct applicability to arbitrary rational functions. We discussed the approach in obtaining suitable coefficients using an algorithmic process combining linear programming and lattice basis reduction. We also discussed design and implementation aspects.

We plan in a future work to improve the efficiency of the linear programming step, both the algorithmic part (some tricks described in [6] could be of use) and the software implementation part (among other things, we would like to avoid using Maple in order to ensure better reliability, time and memory cost and diffusion). A reasonable target, for small degrees (let say 4) of the numerator and denominator, is to try to adapt the results of [5] and to combine them with the results of this paper to obtain the best possible simple E-fraction with fixed-point or floating-point coefficients. We also wish to take into account all the possible scalings [4] in order to improve our results when dealing with larger intervals for instance.

6 Acknowledgements

This work was partially supported by the ACI grant *GAAP* from the French Ministry of Education. This paper was written while the first author was invited by the Center of Mathematics and Applications of the Australian National University, which he warmly thanks for the excellent welcome and conditions of work provided.

References

[1] A. Avizienis. *Signed-digit number representations for fast parallel arithmetic.* IRE Transactions on electronic computers, 10: 389–400, 1961. Reprinted in E.E. Swartzlander, Computer Arithmetic, Vol. 2, IEEE Computer Society Press Tutorial, 1990.

[2] L. Babai, *On Lovász' lattice reduction and the nearest lattice point problem*. Combinatorica, An International Journal of the János Bolyai Mathematical Society, vol. 6, n. 1, 1–13, 1986.

[3] N. Brisebarre and S. Chevillard. *Efficient Polynomial L^∞-Approximations* . 18th IEEE Symposium on Computer Arithmetic (ARITH-18), Montpellier (France), p. 169-176, June 2007.

[4] N. Brisebarre and J.-M. Muller. *Functions approximable by E-fractions*. Proc. 38th Conference on signals, systems and computers, Pacific Grove, California, U.S.A., p. 1341-1344, Nov. 2004.

[5] N. Brisebarre, J.-M. Muller and A. Tisserand. *Computing machine-efficient polynomial approximations*. ACM Trans. Math. Software, Vol. 32, n. 2, 236-256.

[6] E. W. Cheney. *Introdution to Approximation Theory*, AMS Chelsea Publishing, Providence, RI, 1998.

[7] M.D. Ercegovac. *A general method for evaluation of functions and computation in a digital computer*. PhD thesis, Dept. of Computer Science, University of Illinois, Urbana-Champaign, 1975.

[8] M.D. Ercegovac. *A general hardware-oriented method for evaluation of functions and computations in a digital computer.* IEEE Trans. Comp., C-26(7):667–680, 1977.

[9] M.D. Ercegovac and T. Lang. *Digital Arithmetic*, Morgan Kaufmann Publishers - an Imprint of Elsevier Science, San Francisco, 2004.

[10] M.D. Ercegovac and J.-M. Muller. *A Hardware-Oriented Method for Evaluating Complex Polynomials.* Proc . ASAP-07, 2007.

[11] J. F. Hart, E. W. Cheney, C. L. Lawson, H. J. Maehly, C. K. Mesztenyi, J. R. Rice, H. G. Thacher and C. Witzgall, *Computer Approximations*, Robert E. Krieger Publishing Company, Florida, 1978.

[12] A. K. Lenstra, H. W. Lenstra and L. Lovász, *Factoring Polynomials with Rational Coefficients*, Math. Annalen, vol. 261, 515–534, 1982.

[13] I. Koren and O. Zinaty. *Evaluating elementary functions in a numerical coprocessor based on rational approximations.* IEEE Trans. Comp., 39(8):1030–37, 1990.