A GENERAL METHOD FOR EVALUATION OF FUNCTIONS AND COMPUTATIONS IN A DIGITAL COMPUTER

Miloš D. Ercegovac*
Computer Science Department
School of Engineering and Applied Science
University of California
Los Angeles, California 90024

Summary

This paper presents a recently discovered general computational method, amenable for efficient implementation in digital computing systems. The method provides a unique, simple and fast algorithm for solving many computational problems, such as the evaluation of polynomials, rational functions and arithmetic expressions, or solving a class of systems of linear equations, or performing the basic arithmetics. In particular, the method is well suited for fast evaluation of commonly used mathematical functions.

Introduction

The subject of this paper is a novel computational method, described in detail elsewhere.[1] The principal idea of the proposed method is based on the recognition of the importance of the following issues in the design of fast algorithms for numerical computations. The first issue is concerned with the compatibility of the problem representation with the implementation environment. Many well-known methods of computations may seem optimal from a human point of view but this can be hardly true when, for example, a digital-type hardware must be used for implementation. The second issue relates the redundancy in the number representation system to the complexity and the speed of the algorithms and the associated primitive operators.

When a computational method is being considered for possible hardware-level implementation, one is usually concerned with i) its application domain, ii) the required set of algorithms, and iii) the required set of primitive operators. With these, one also associates a set of properties, for instance, the speed, the complexity and the cost of implementation, numerical characteristics of the algorithms, etc. Ideally, an implemented computational method should have an unrestricted domain of applications, a single but simple algorithm to be performed and only those primitive operators which are efficiently realizable in the given implementation technology. The proposed method comes close in satisfying these design objectives. It also provides, we believe, a highly potential alternative in the design of numerical algorithms compared to both the parallel techniques, utilizing a multiplicity of general-purpose processors and the strictly hardware-oriented algorithms.

One of the original motivations of this work was the problem of fast and efficient evaluation of commonly used mathematical functions. The method evolved while attempting to solve this problem with the above mentioned objectives in mind: an open domain of application, as in function evaluation methods, based on the classical approximation techniques[2] with the performance of the hardware-oriented algorithms[3-10].

The computational algorithm of the proposed method shares some properties with the incremental computations in the digital differential analysers (DDA) although it is not based on an integration principle. The potential of such an algorithm in implementing multiprocessing systems has been recognized also by Campeau[11,12,13].

The Computational Method

In this section a general evaluation technique, named E-method for brevity, is defined in general terms as:

(i) A problem-dependent correspondence rule C_f, which associates independent variables $\underline{x}_f$ dependent variables $\underline{y}_f$, and parameters $\underline{p}_f$ of a given computational problem $f(\underline{x}_f, \underline{p}_f)$ with a system L of simultaneous linear equations $\underline{A}_f \underline{y} = \underline{b}_f$ in such a way that there is a one-one correspondence between dependent variables $\underline{y}_f$, i.e., the results of f, and the solution $\underline{y}$ of the system L. The elements of the matrix $\underline{A}_f$ and vector $\underline{b}_f$ must satisfy certain conditions, as specified later;

(ii) A problem independent algorithm for solving the system L in time linearly proportional to the desired number of correct digits of the solution $\underline{y}$, and which is amenable to an efficient implementation.

A computational problem f is said to be L-reducible if there exists a correspondence rule C_f, not necessarily unique. The E-method is applicable in all L-reducible problems with an equal efficiency: the computational algorithm remains invariant while the particular correspondence rule, no more complex to perform than the assignment of values, characterizes the problem.

The correspondence rules will be illustrated later in several application examples, while the algorithm is given next.

The Computational Algorithm

The algorithm is defined in such a way that the following, for an efficient implementation important properties hold:

a) The algorithm generates the most significant digits of the result $\underline{y}$ first in such a way that once generated digit, y_j, at the step j will not be affected by any subsequent step k, $k<j$;

b) The basic computational step is invariant with addition as the only primitive operator; the selection procedure which generates one digit of the result per step is deterministic and feasible on a limited argument precision

*This work was done while the author was with the Department of Computer Science of the University of Illinois at Urbana-Champaign under the support of grant No. US NSF DCR 73-07998 and the Department of Computer Science.

so that the step execution time is independent of the length of operands;

c) The algorithm has an "on-line" capability with respect to the independent variable. Namely, if $\{x_j \mid i = 1,\ldots,m\}$ are the digits of the independent variable x then only the digit x_j need be used at the $(j+1)$ st step.

Definition 1: An m digit radix r representation of a number x, $|x| < 1$, is a polynomial expansion

$$x = \text{sign } x \cdot \sum_{i=1}^{m} x_i \, r^{-i}$$

where

$$x_i \in D, \quad \forall i$$

and D is a given digit set.

Definition 2: For a given radix r, a set of consecutive integers D is

i) a nonredundant digit set if its cardinality satisfies

$$|D| = r$$

ii) a redundant digit set if

$$|D| > r$$

Definition 3: A symmetric redundant digit set is defined as

$$D_\rho = \{-\rho,-(\rho-1),\ldots,-1,0,1,\ldots,\rho-1,\rho\}$$

where

$$\frac{r}{2} \leq \rho \leq r - 1.$$

In particular, D_ρ is

i) minimally redundant if

$$|D_\rho| = r + 1$$

so that

$$\rho = \frac{r}{2},$$

assuming an even radix r;

ii) maximally redundant if

$$|D_\rho| = 2r - 1$$

so that

$$\rho = r - 1 .$$

Consequently, the representation of a number x is redundant or nonredundant depending whether $x_i \in D_\rho$ or $x_i \in D$. In the case of a redundant representation

$$\text{sign } x \equiv \text{sign } x_1$$

so $x = \sum_{i=1}^{m} x_i \, r^{-i}$.

Let n be the order of the system L and $j = 1,2,\ldots,m+1$ the recursion index. Define: the j-th digit vector $\underline{d}^{(j)}$ as

$$\underline{d}^{(j)} = [d_1^{(j)}, d_2^{(j)}, \ldots, d_n^{(j)}] ;$$

the j-th residual vector $\underline{z}^{(j)}$ as

$$\underline{z}^{(j)} = [z_1^{(j)}, z_2^{(j)}, \ldots, z_n^{(j)}]$$

and their vector sum $\underline{w}^{(j)}$ as

$$\underline{w}^{(j)} = \underline{d}^{(j)} + \underline{z}^{(j)} = [w_1^{(j)}, w_2^{(j)}, \ldots, w_n^{(j)}]$$

where

$$w_i^{(j)} = d_i^{(j)} + z_i^{(j)} , \quad i = 1,2,\ldots,n , \quad \forall j.$$

Let $s(\underline{w}(j))$ be the vector selection function

$$s(\underline{w}^{(j)}) = [s(w_1^{(j)}), s(w_2^{(j)}), \ldots, s(w_n^{(j)})]$$

such that

$$d_i^{(j)} = s(w_i^{(j)}) , \quad i = 1,2,\ldots,n , \quad \forall j$$

and the selection function is defined as follows:

$$d_i^{(j)} = s(w_i^{(j)}) = \begin{cases} \text{sign } w_i^{(j)} \left\lfloor |w_i^{(j)}| + 1/2 \right\rfloor \\ \quad \text{-for } |w_i^{(j)}| \leq \rho \\ \text{sign } w_i^{(j)} \left\lfloor |w_i^{(j)}| \right\rfloor \\ \quad \text{-otherwise} \end{cases} \tag{1}$$

where $\lfloor w \rfloor$ denotes the integer part of w. The selection function $s(w_i^{(j)})$ is, for practical purposes, represented as a rounding procedure, modified at the endpoints of the domain to avoid digit values $|d_i^{(j)}| > \rho$. It simply maps a w-subinterval $[k-\frac{1}{2}, k+\frac{1}{2})$ to an integer k. Furthermore, define the n-th order matrix $\underline{G}(x)$ as

$$\underline{G}(x) = \underline{I} - \underline{A}(x) = (g_{ij})_{n \times n} \tag{2}$$

where $\underline{I}$ is the identity matrix and $\underline{A}(x)$ is the nonsingular coefficient matrix of the system L defined by the correspondence rule C_f. The maximum vector norm

$$\|\underline{x}\|_\infty = \max_i |x_i| \tag{3}$$

and the consistent matrix norm

$$\|\underline{A}\|_\infty = \max_i \left(\sum_{j=1}^{n} |a_{ij}| \right), \tag{4}$$

as the only norms considered, will be denoted as $\|\underline{x}\|$ and $\|\underline{A}\|$, respectively.

Theorem 1

If

$$\underline{G}(x) \leq \alpha \tag{5}$$

where the bound α satisfies

$$0 < \alpha \leq 1/r\left[1 - \frac{\zeta(r-1)}{\rho}\right]; \tag{6}$$

$$\|\underline{b}\| \leq \zeta \tag{7}$$

where the bound ζ satisfies

$$1/2 \leq \zeta < 1 \tag{8}$$

and

$$d_i^{(j)} \in D_\rho, \forall_i \ \forall_j$$

then the following n-th order system of linear recursions

$$\underline{w}^{(j)} = \underline{d}^{(j)} + \underline{z}^{(j)} = r(\underline{z}^{(j-1)} + \underline{G}(x)\, \underline{d}^{(j-1)}), \tag{9}$$

$$j = 1,2,\ldots,m+1$$

with the initial conditions

$$\underline{d}^{(0)} = 0 ,$$

$$\underline{z}^{(0)} = \underline{b} , \tag{10}$$

generates m leftmost correct digits of the solution

$$\underline{y} = \underline{A}^{-1}(x)\, \underline{b}$$

in at most m+1 steps as the sequence $\{\underline{d}^{(j)}\}$, $\underline{d}^{(j)}$, $\forall_j$, selected according to (1). Namely, the generated solution

$$\underline{y}^* = \sum_{j=1}^{m+1} \underline{d}^{(j)} r^{-j} = [\sum_{j=1}^{m+1} d_1^{(j)} r^{-j}, \ldots, \sum_{j=1}^{m+1} d_n^{(j)} r^{-j}] \quad (11)$$

satisfies

$$\|\underline{y} - \underline{y}^*\| < r^{-m} . \quad (12)$$

Proof:

The consistency of the system of recursions with respect to the selection function (1) is proved inductively. By the statement of the theorem,

$$\|\underline{z}^{(0)}\| \leq \zeta .$$

Assume that

$$\|\underline{z}^{(j-1)}\| \leq \zeta .$$

Then

$$\begin{aligned} \|\underline{w}^{(j)}\| &= \|\underline{d}^{(j)} + \underline{z}^{(j)}\| \quad (13) \\ &= r\|\underline{z}^{(j-1)} + \underline{G}(x)\, \underline{d}^{(j-1)}\| \\ &\leq r\|\underline{z}^{(j-1)} + r\, \underline{G}(x) \cdot \underline{d}^{(j-1)}\| \\ &\leq r\,\zeta + r\,\alpha\,\rho \\ &= r\,\zeta + r[\tfrac{1}{r}(1 - \tfrac{\zeta(r-1)}{\rho})]\rho \\ &= \rho + \zeta . \end{aligned}$$

Since

$$\underline{z}^{(j)} = \underline{w}^{(j)} - \underline{d}^{(j)} \text{ and sign } d_i^{(j)} = \text{sign } w_i^{(j)} ,$$

by definition of the selection function $s(\underline{w}^{(j)})$, it immediately follows that

$$\|\underline{z}^{(j)}\| \leq \zeta .$$

The convergence is proved by showing that the solution error vector

$$\underline{h} = [h_1, h_2, \ldots, h_n] = \underline{y} - \underline{y}^* \quad (14)$$

satisfies

$$\|\underline{h}\| < r^{-m} . \quad (15)$$

After m+1 steps the following holds:

$$\begin{aligned} \underline{d}^{(m+1)} + \underline{z}^{(m+1)} &= r^{m+1}\, \underline{b} + \underline{G}(x)\, [\sum_{j=1}^{m} \underline{d}^{(j)} r^{m+1-j}] \\ &\quad - \underline{I}[\sum_{j=1}^{m} \underline{d}^{(j)} r^{m+1-j}] \end{aligned} \quad (16)$$

or

$$r^{-m-1}\, \underline{z}^{(m+1)} = \underline{b} - \underline{A}(x) \cdot \underline{y}^* - \underline{G}(x)\, \underline{d}^{(m+1)} r^{-m-1} .$$

Let

$$\underline{e}^{(m+1)} = [e_1^{(m+1)}, e_2^{(m+1)}, \ldots, e_n^{(m+1)}]$$

$$= (\underline{z}^{(m+1)} + \underline{G}(x)\underline{d}^{(m+1)}) r^{-m-1} .$$

Since $\underline{y} = \underline{A}^{-1}(x)\, \underline{b}$,

$$\underline{h} = \underline{y} - \underline{y}^* = \underline{A}^{-1}(x) \cdot (-\underline{e}^{(m+1)}) . \quad (17)$$

The error after m+1 steps is, therefore, bounded by

$$\begin{aligned} \|\underline{h}\| &= \|\underline{A}^{-1}(x)(-\underline{e}^{(m+1)})\| \quad (18) \\ &\leq \|\underline{A}^{-1}(x)\| \cdot \|\underline{e}^{(m+1)}\| . \end{aligned}$$

Since $\underline{G}(x) \leq \alpha < 1$ by definition for $r \geq 2$, the matrix $\underline{G}(x)$ is convergent, i.e.,

$$\lim_{p \to \infty} [\underline{G}(x)]^p = 0$$

By the well-known result[18]

$$\begin{aligned} \|\underline{A}^{-1}(x)\| &= \|\underline{I} + \underline{G}(x) + \underline{G}^2(x) + \ldots\| \\ &\leq \|\underline{I}\| + \|\underline{G}(x)\| + \|\underline{G}(x)\|^2 + \ldots \\ &\leq 1 + \sum_{p=1}^{\infty} \alpha^p \\ &= \frac{1}{1-\alpha} \leq \frac{4}{3} \end{aligned} \quad (19)$$

Also

$$\begin{aligned} \|\underline{e}^{(m+1)}\| &= r^{-m-1}\|\underline{z}^{(m+1)} + \underline{G}(x)\, \underline{d}^{(m+1)}\| \\ &\leq r^{-m-1}(\zeta + \alpha\rho) . \end{aligned} \quad (20)$$

Therefore,

$$\|\underline{h}\| \leq \frac{1}{1-\alpha} \cdot \frac{\zeta+\alpha\rho}{r} \cdot r^{-m} = \gamma \cdot r^{-m} \quad (21)$$

where

$$\gamma = \frac{1}{2(r-1)} \quad (22)$$

for minimally redundant digit set and

$$\gamma = \frac{1}{r} \quad (23)$$

for maximally redundant digit set. Since $\gamma < 1$ for $r > 1$

$$\|\underline{h}\| < r^{-m} \qquad \square$$

Theorem 1 completes the general specification of the computational part of the E-method. Clearly, the recursion formula may be computed in time independent of the operands precision, provided that the approximate, low-precision value $\hat{\underline{w}}^{(j)}$ of the sum $\underline{w}^{(j)}$ satisfies the selection requirement

$$\|\underline{w}^{(j)} - \hat{\underline{w}}^{(j)}\| \leq \frac{\Delta}{2} \quad (24)$$

where Δ is the given overlap between the selection subintervals [1] as illustrated in Figure 1.

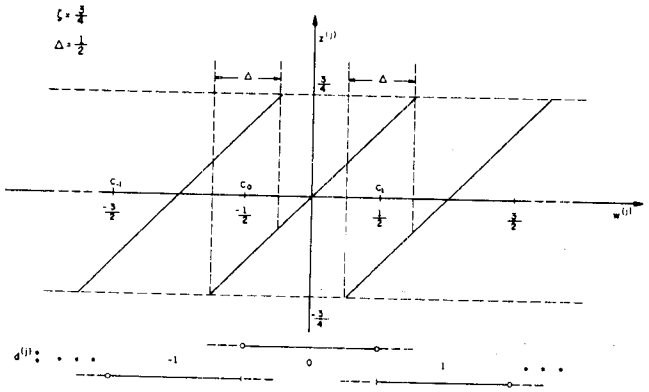

Figure 1. Limited Precision Selection Procedure

The relationships between the selection overlap Δ and the bounds α, ζ and ρ are summarized in the following table:

Mode of Operation:	Redundancy in D_f: Minimal $\rho = \frac{r}{2}$	Redundancy in D_f: Maximal $\rho = r - 1$
Operand Precision Dependent $\zeta = \frac{1}{2}$ $\Delta = 0$	$\alpha \leq \frac{1}{r}$	$\alpha \leq \frac{1}{2r}$
Operand Precision Independent $\zeta = \frac{1}{2}(1+\Delta)$ $0 < \Delta < \frac{2\rho}{r-1} - 1$	$\alpha \leq \frac{1}{r^{\Delta}}[1-\Delta(r-1)]$	$\alpha \leq \frac{1}{2r}[1-\Delta]$

Table 1. Relationships Between Bounds

Error Properties

The error behavior of the E-method appears to be quite favorable. It can be seen that if the system L satisfies the bounds, as required by Theorem 1, then it is insensitive to small perturbations in elements of $\underline{A}(x)$ and $\underline{b}$ since the corresponding condition number of the system's matrix satisfies

$$K(\underline{A}(x)) = \|\underline{A}(x)\| \cdot \|A^{-1}(x)\| = \frac{1+\alpha}{1-\alpha} \leq \frac{5}{3}.$$

Moreover, by definition of the computational algorithm, no roundoff errors are generated when E-method is applied. The inevitable effects of a finite precision number representation[1] are of minor consequences.

Scaling

The conditions (5) and (7) of Theorem 1 on norms of matrix $\underline{A}(x)$ and vector $\underline{b}$, imply that, in general, an adjustment of the size of the elements a_{ij} and b_i will be required. Although scaling commonly appears whenever fixed-point representation arithmetic is used, it can be handled without serious difficulties. The E-method, however, requires more consideration of the scaling problem[1], but the details, for brevity, are omitted here. We note that in many applications, scaling can be easily performed while in certain cases the scaling requirements can be accomodated by redefining the problem.

Summary of the E-method

The E-method is summarized below for reference convenience:

Part 1 (Correspondence)

Given an L-reducible computational problem f, apply the correspondence C_f on the arguments and parameters of f in order to obtain the coefficient matrix $\underline{A}$ and the vector $\underline{b}$ of the system of linear equations L. The correspondence rule C_f must guarantee that the elements of $\underline{A}$ and $\underline{b}$ can be made conformable to the conditions:

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \leq \alpha, \quad \forall i$$

$$|b_i| \leq \zeta, \quad \forall i$$

Part 2 (Computation)

Algorithm E

/ Initialization /

1. $\underline{z}^{(0)} \leftarrow \underline{b}$; $\underline{d}^{(0)} \leftarrow 0$;

/ Recursion /

2. for j = 1,2,...,m+1:

2.1 $\underline{w}^{(j)} \leftarrow r(\underline{w}^{(j-1)} - \underline{A}(x)\underline{d}^{(j-1)})$;

2.2 $\underline{d}^{(j)} \leftarrow s(\underline{w}^{(j)})$;

/ Termination /

3. HALT

/ The result(s) of f, for the given precision m, are represented by

$$\underline{y}^* = \sum_{j=1}^{m+1} \underline{d}^{(j)} r^{-j}$$

$$= [y_1^*, y_2^*, \ldots, y_n^*] \quad /$$

On Implementation and Performance of the E-Method

The basic computational block, the elementary unit EU_i, is a hardware structure implementing the basic recursion formula (9) or, more precisely, the recursion step of the Algorithm E. It seems preferable, from the implementation point of view, to restate the basic recursion formula (9) as follows:

$$w_i^{(j)} = r(w_i^{(j-1)} + \sum_{k=1}^{n} g_{ik} d_k^{(j-1)}) \qquad (25)$$

where $g_{ij} = -1$, so that the need for explicitly calculated residuals $z_i^{(j)}$ is avoided.

As indicated by Figure 2 where a global structure of the elementary unit EU_i is shown, the evaluation of $w_i^{(j)}$ basically requires an s-operand adder. In many practical applications s is rather small. In order for the time of addition to be independent of operand precision, $w_i^{(j)}$ need to be represented in a redundant form.

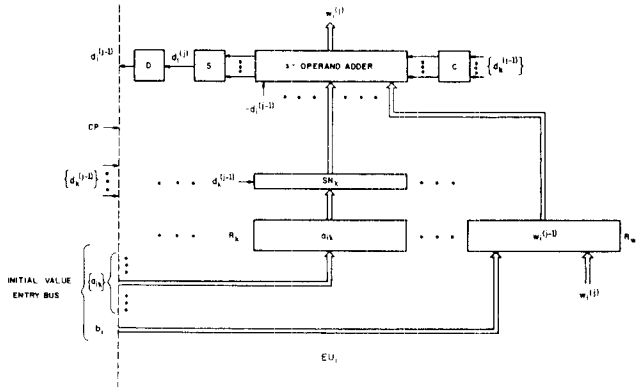

Figure 2. Elementary Unit Structure

The selection procedure, defined by the selection function $s(\hat{w}_i^{(j)})$ is performed by the block S, which forms $\hat{w}_i^{(j)}$ by converting a few of the most significant digits of $w_i^{(j)}$ into nonredundant form. After rounding $\hat{w}_i^{(j)}$, the integer part represents the selected digit $d_i^{(j)}$. The precision of $\hat{w}_i^{(j)}$ is basically determined by the overlap Δ and the number s of summands. All functions of the selection block S can be easily implemented. The previous value $d_i^{(j-1)}$ is saved in register D. The coefficients $\{a_{ik}\}$ may be, for better storage efficiency and simpler adder structure, represented in a nonredundant form. The single, signed digit radix-r multipliers $d_k^{(j)}$ are incorporated through the selection networks $\{SN_k\}$, each capable of forming required multiples of a_{ik}. The carry generator C would be needed if, for example, a radix complement representation of negative numbers is adopted. In that case, the selection networks merely form direct or complement of a possibly shifted value of a_{ik}. The complexity of the selection networks increases for higher radices, and since the additional multiples appear as summands, complexity of the adder will also be increased. Therefore, a higher radix, while reducing the necessary number of steps for a given precision, does increase both the time to perform the basic recursion and the complexity of the corresponding elementary unit.

The central part of an EU, the multioperand adder, can be implemented in various ways in order to achieve the desired speed/cost factor. The registers R_k store the corresponding coefficients a_{ik} throughout a particular evaluation; their number for the elementary unit EU_i is determined by the number of nonzero elements in the i-th row of the matrix $\underline{G}(x)$, i.e., the number of inputs to EU_i. Register R_w and the corresponding data path must accommodate the redundantly represented $w_i^{(j)}$. The initialization, i.e., the execution of the particular correspondence rule C_f is performed by loading the coefficients $\{a_{ik}\}$ and b_i via the initial value entry bus.

The control requirements of an EU are very simple: assuming a synchronous mode of operation of the entire configuration for the elementary units, synchronizing pulses on which the transfer of $w_i^{(j)}$ into R_w occurs, are all that is needed. The same clock pulses, defining the basic step, are distributed to all units. By controlling their number, one can easily achieve a variable precision mode of operation.

The time required to perform one recursive step on an elementary unit is defined as:

$$t_0 = t_A + t_S + t_T$$

where t_A is the time to generate $w_i^{(j)}$ in a redundant form, t_S is the selection time and t_T is the register transfer time. Both t_S and t_T correspond to a few gate delays--(3-4) t_g, so that t_A appears as the dominant factor, which depends on the number s of summands and the adder structure. For practical reasons, s may be defined to denote the number of radix-2 summands, i.e., the higher radix or redundantly represented operands are replaced by their binary equivalents. Then a simple adder structure, consisting of s-2 levels of full-adder rows, will have a $t_A = 2(s-2)\ t_g$, assuming $2t_g$ per full-adder. More sophisticated adder structures,[14] i.e., Dadda-type can considerably reduce this time. However, when s is small, like in polynomial evaluation, it can be seen that, for radix 2, $t_0 = O(10t_g)$.

A Graph Representation of a General Computing Configuration

An L-reducible problem f of order n can be solved by the E-method on a structure consisting of n interconnected elementary units. The computational algorithm E indicates that the intercommunication requirements are simple due to the fact that the elementary units are functionally related to each other only via the digit vector $\underline{d}$. This implies that the physical connection between the units EU_i and EU_j only needs to accommodate a transfer of one, signed radix r digit. In common multiprocessor structures, used for fast parallel computations, the processor intercommunications usually require full precision width.

A computing structure for solving a given problem f by the E-method may be conveniently specified by a computational graph $G_f(V,\underline{K})$ where $V = \{V_i \mid i=1,\ldots,n\}$ is a set of vertices and $\underline{K}$ is a connection matrix, defining a set of directed arcs. Each vertex V_i corresponds to an elementary unit EU_i, symbolically represented as in Figure 3 where the outgoing arc d_i carries the digit generated by EU_i and one or more incoming arcs d_j, inputs to EU_i, carry the digits generated by $\{EU_j\}$. The connection matrix $\underline{K} = (k_{ij})_{n\times n}$ is in one-one correspondence with the matrix $\underline{G}(x) = \underline{I} - \underline{A}(x)$ of the system L as follows:

$$k_{ij} = \begin{cases} 1 & \text{if } g_{ij} \neq 0 \\ 0 & \text{if } g_{ij} = 0 \end{cases} \qquad (26)$$

and $k_{ij} = 1$ specifies that the arc d_j is an incoming arc for the vertex V_i, i.e., the connection matrix $\underline{K}$ defines the relation "receives from" between the elementary units $\{EU_i\}$ of the computing structure. Note

that the utilization of d_i for internal functions in EU_i, as indicated in Figure 2 need not be specified on the graph unless $g_{ii} \neq 0$.

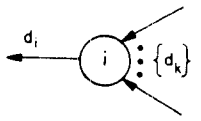

Figure 3. Elementary Unit: Graph Representation

The Computational graph G_f may be acyclic or cyclic. The graph G_f is acyclic if and only if its connection matrix $\underline{K}$ is strictly upper (lower) triangular. In a practical sense, a cyclic graph G_f corresponds to a problem f which involves division.

Importantly, a graph G_f provides a direct estimate of the required time and implementation complexity. If $N(\underline{K})$ denotes the number of ones in the connection matrix $\underline{K}_{n\times n}$ then the computational structure requires at most $N(\underline{K})$ + 2n m-digit registers, n adders with total of $N(\underline{K})$ + 2n operands and $N(\underline{K})$ single digit interconnections. It is assumed that two registers are sufficient to store w_i value in a redundant form. If $N_i = |\{d_k\}|$ denotes the number of inputs to the i-th elementary unit, then EU_i requires $s = (N_i+2)$ operand adder and that many registers.

The time t_0 required to perform one recursive step is clearly determined by the parameters of the elementary unit EU_k such that $N_k = \max_i (N_i)$. The total time, then, for the E-method evaluation, assuming an m-digit precision, is

$$T_E(f) = (m+1)\ t_0$$

if no scaling is required, and

$$T_E(f) = (m+1+\sigma)\ t_0$$

otherwise, where integer $\sigma > 0$ is defined by the particular scaling requirements.

We conclude this section with the following remarks. The functional properties of the E-method, namely the step-invariant nature of its computational algorithm and linearity of its primitive operator, make an adaptation both to a variable precision mode of operation and a variable number of elementary units rather straightforward. We mention also that the "on-line" capability of the E-method can be particularly important in real-time applications.

On Applications of the E-method

We begin by describing the evaluation of polynomial and rational polynomial functions of a single variable. It can be seen that any polynomial is L-reducible, which is not true for rational functions. However, L-reducibility of a rational function can be assured by taking into account the appropriate conditions when the coefficients of the rational function are being defined.

While considering problems of evaluating polynomials and rational functions, we will attempt to provide relative measures of performance with respect to a conventional sequential or S-method and a parallel or P-method of computation. We will use the speed-up S, efficiency E and cost C factors as defined by Kuck[15], but under different assumptions. Namely, as a matter of implementation complexity standardization, we will assume that a processor used by S-, or P-method is equivalent in its capability and complexity to an elementary unit of the E-method. We will also consider all three methods in a domain of operations on a hardware level so that the usual assumption that each arithmetic operation takes the same unit time does not hold here. The relative performance measures for the other application examples can be derived in a similar way and are omitted here.

Evaluation of Polynomials

Let

$$P_\mu(x) = \sum_{i=0}^{\mu} p_i\, x^i \tag{27}$$

be a μ-degree polynomial with real-valued coefficients $\{p_i\}$ and the argument $x \in [a,b]$.

Assume that $\underline{p}$ and $\underline{x}$ are vectors with p_i's and all representable values of $x \in [a,b]$ as the components, respectively.

If

$$\|\underline{p}\| \leq \zeta$$

and

$$\|\underline{x}\| \leq \alpha \tag{28}$$

then the problem of evaluating the polynomial $P_\mu(x)$ is immediately reducible to the system L: $\underline{A}(x)\ \underline{y} = \underline{b}$ of order $n = \mu+1$ according to the following correspondence rule C_p:

$$a_{ij} = \begin{cases} 1 & \text{for } i=j; \\ -x & \text{for } j=i+1,\ i \leq \mu; \\ 0 & \text{otherwise}; \end{cases}$$

$$b_i = \begin{cases} p_{i-1} & \text{for } i=1,2,\ldots,\mu+1; \\ 0 & \text{otherwise}, \end{cases} \tag{29}$$

as illustrated in Figure 4. The system L is then solved using the Algorithm E so that after m+1 steps

$$\underline{y}^* = \sum_{j=1}^{m+1} \underline{d}^{(j)}\, r^{-j} \tag{30}$$

satisfies

$$|P_\mu(x) - y_1^*| < r^{-m} \tag{31}$$

and

$$\left| \sum_{k=i-1}^{\mu} p_k\, x^{k-i+1} - y_i^* \right| < r^{-m} \tag{32}$$

for $i = 2,3,\ldots,\mu+1$.

The computational graph G_p is shown in Figure 5. All the elementary units EU_i, $i=1,2,\ldots,n$ are identical: each one requires an adder with one redundant $(w_i^{(j)})$ and one nonredundant $(x.d_{i+1}^{(j)})$ operand, or, effectively, a conventional adder, where r=2.

$$\begin{bmatrix} 1 & -x & & & & \\ & 1 & -x & & & \\ & & 1 & -x & & \\ & & & \ddots & \ddots & \\ & & & & 1 & -x \\ & & & & & 1 \end{bmatrix} \times \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_\mu \end{bmatrix}$$

Figure 4. Correspondence Rule C_P

In general the conditions on norms (28) may not be satisfied and an appropriate scaling of p_i's and x must be done prior to the evaluation. The required scaling in this case presents no problems.[1]

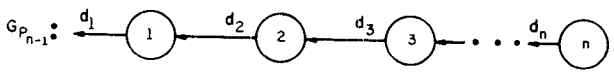

Figure 5. Computational Graph G_p

We now consider the performance of the E-method in polynomial evaluation relative to the S-, and P-methods. Both these methods are assumed to be using an iterative multiplication algorithm so that the basic processing unit in all three methods can be considered equivalent in complexity and speed. Furthermore, we assume a fixed-point representation domain with no scaling requirements. Denoting addition time as t_0, we have the evaluation times and the number of processors for E-, S-, and P-method as follows:

$$\begin{aligned} T_E(P_\mu) &= (m+1)\, t_0,\ n_E = \mu + 1 \\ T_S(P_\mu) &= \mu\, m\, t_0,\ n_S = 1 \\ T_P(P_\mu) &= (\ell og_2\mu + (O(\ell og_2\mu)^{1/2})m)\, t_0,\ n_P = 2\mu \end{aligned} \tag{33}$$

We have assumed that the P-method uses Maruyama-Munro-Paterson algorithm for parallel polynomial evaluation[16].

Following Kuck[15], the E-method algorithm for polynomial evaluation has the speedup factor S_E:

$$S_E = \frac{T_S}{T_E} = \mu\left(\frac{m}{m+1}\right) \approx \mu\ , \tag{34}$$

the efficiency factor E_E:

$$E_E = \frac{S_E}{N_E} = \left(\frac{\mu}{\mu+1}\right)\left(\frac{m}{m+1}\right) > 0.5 \text{ for } \mu > 1,\ m > 3 \tag{35}$$

and the cost factor C_E:

$$C_E = n_E \cdot T_E = (\mu+1)(m+1)\, t_0 \tag{36}$$

Similarly for the P-method algorithm:

$$S = \frac{T_S}{T_P} \approx \frac{\mu}{\sqrt{M}} \cdot \frac{m}{\sqrt{M}+m} \approx \frac{\mu}{\sqrt{M}}$$

$$E_P = \frac{S_P}{n_P} \approx \frac{1}{2\sqrt{M}} \cdot \frac{m}{\sqrt{M}+m} \tag{37}$$

$$C_P \approx 2\mu\ (M + \sqrt{M}\cdot m)\ t_0$$

where $M = \ell og_2\mu$. As a reasonable measure of performance, Kuck suggests the ratio of effectiveness, given by speedup, and cost of the method so that the A-method is better in performance than the B-method if

$$\max\left(\frac{S_A}{C_A}, \frac{S_B}{C_B}\right) = \frac{S_A}{C_A}. \tag{38}$$

It can be easily seen that

$$\lim_{\mu\to\infty} \frac{S_E}{C_E} = \frac{m}{(m+1)^2 t_0} \approx \frac{1}{mt_0}$$

while

$$\lim_{\mu\to\infty} \frac{S_P}{C_P} = 0 \tag{39}$$

With respect to the precision, both $\lim_{m\to\infty} \frac{S_E}{C_E} = 0$ and $\lim_{m\to\infty} \frac{S_P}{C_P} = 0$ but

$$\lim_{m\to\infty} \frac{S_E}{C_E} \cdot \frac{C_P}{S_P} = \frac{2\mu\ell og_2\mu}{\mu+1} > 1 \text{ for } \mu > 1. \tag{40}$$

This indicates that the E-method algorithm is better in performance noticeably than any P-method algorithm for the evaluation of polynomials under the previously stated conditions. Furthermore, the E-method algorithm has a behavior of performance measure qualitatively different from the considered P-method algorithm, as illustrated in Figure 6. In this example it is assumed that m = 56, r = 2 and, for presentation convenience, that $t_0 = 10^{-2}$ units.

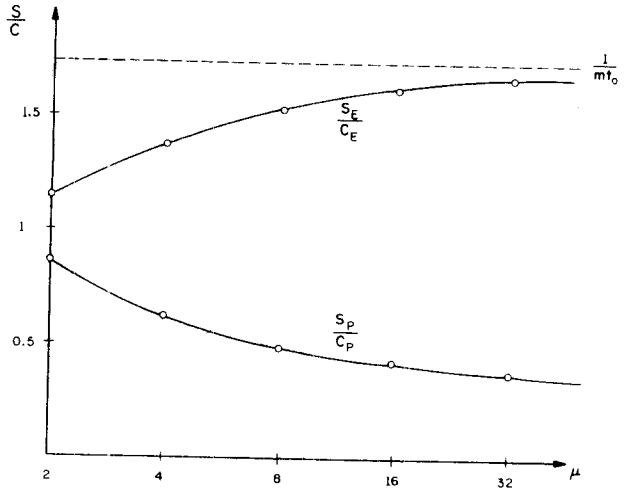

Figure 6. Evaluation of Polynomials: Performance Measures

Evaluation of Rational Functions

Let $R_{\mu,\nu}(x)$ be a real-valued rational function:

$$R_{\mu,\nu}(x) = \frac{P_\mu(x)}{Q_\nu(x)} = \frac{\sum_{i=0}^{\mu} p_i x^i}{\sum_{i=0}^{\nu} q_i x^i} \qquad (41)$$

Without loss of generality it is assumed that $q_0=1$. Let

$$\underline{A}(x)\ \underline{y} = \underline{b} \qquad (42)$$

be a nonhomogeneous system of n simultaneous linear equations.

Theorem 2

If $\max(\mu,\nu) \leq n-1$ and the coefficients a_{ij}'s, b_i's of the system (42) are put into correspondence with the coefficients p_i's, q_i's and the argument x according to the following rule C_R:

$$a_{ij} = \begin{cases} 1 & \text{for } i = j; \\ q_{i-1} & \text{for } j = 1 \text{ and } i = 2,3,\ldots,\nu+1; \\ -x & \text{for } j = i+1 \text{ and } i = 1,2,\ldots,n-1; \\ 0 & \text{otherwise;} \end{cases}$$

$$b_i = \begin{cases} p_{i-1} & \text{for } i = 1,2,\ldots,\mu+1 \\ 0 & \text{otherwise,} \end{cases}$$

then

$$y_1(x) = \frac{D_1(x)}{D(x)} = \frac{P_\mu(x)}{Q_\nu(x)} = R_{\mu,\nu}(x)\ .$$

Proof:

By the Laplace expansion of the determinants. □

The correspondence rule C_R defines a linear system L: $\underline{A}(x)\ \underline{y} = \underline{b}$ for a given L-reducible rational function $R_{\mu,\nu}(x)$. Algorithm E can be carried out on a configuration represented by the graph G_R with $n = \max(\mu,\nu) + 1$, as illustrated in Figure 7.

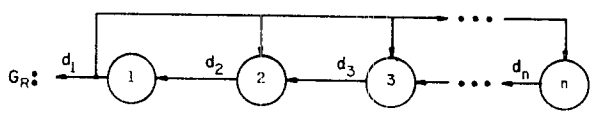

Figure 7. Computational Graph G_R

The adder structure of the elementary unit needs to accommodate at most two nonredundant and one redundant operand and that, for radix 2, can be easily achieved with a two-level conventional adder. The conditions under which a rational function is L-reducible are considered in detail elsewhere.[1] An example, given in Appendix, illustrates the evaluation of rational functions.

Let us consider the performance of the E-method in evaluating rational functions under the same assumptions as in the previous section. The sequential S-method has

$$T_S(R_{\mu,\nu}) \approx (\mu+\nu+1)\ \frac{m}{2}\ t_0\ ,$$

assuming that two digits of the multiplier, or the quotient at the end of the evaluation, can be retired at each step. Since

$$T_E(R_{\mu,\nu}) = (m+1)\ t_0$$

the speed-up of the E-method is

$$S_E = \frac{T_S}{T_E} \approx \frac{\mu+\nu+1}{2}$$

with an efficiency

$$E_E = \frac{S_E}{n} \approx \frac{\mu+\nu+1}{2(\max(\mu,\nu)+1)}$$

For $\mu=\nu$, $E_E \geq 0.75$. We know of no P-method algorithm for which

$$T_P(R_{\mu,\nu}) \leq T_P(P_{\max(\mu,\nu)})$$

and

$$E_P(R_{\mu,\nu}) \leq E_P(P_{\max(\mu,\nu)})$$

so, on the basis of the conclusions made in the previous section, we may again conclude that the E-method is faster and more efficient than a P-method, under the stated assumptions.

Evaluation of Elementary Functions

With a capability to efficiently evaluate polynomial and rational functions, the E-method can certainly be used for the evaluation of arbitrary functions for which suitable polynomial or rational approximations exist. Hence, the evaluation of a given function would be characterized mainly by the corresponding set of coefficients, which can be kept in a local storage area of the computing configuration, or on any other convenient level in the available storage hierarchy. Furthermore, the evaluation could be performed easily in a variable precision. In general, given a sufficient number of the two-input elementary units, an evaluation of a polynomial approximation would take $T_E(f) = O(10m)\ t_g$, where t_g is a gate delay and the radix is binary. Since the relationship between the speed and the cost of the E-method is linear, a wide range of evaluation requirements can be easily accomodated. It may be of interest to mention that Algorithm E can be easily adapted for fast multipoint evaluation of a given function.

On Performing the Basic Arithmetics

The basic recursion of the Algorithm E can clearly be used for additions, subtractions or multiplications. The result obtained will be in a redundant form. By considering division as an L-reducible problem[1], Algorithm E can be used to generate the quotient and the remainder in a deterministic fashion[17], with basic recursive step executable in time independent of the length of the operands. The computational graphs for division and multiplication are shown in Figure 8.

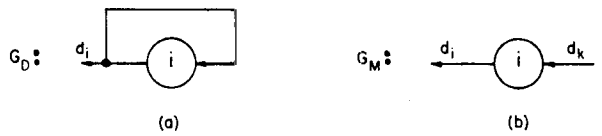

Figure 8. Computational Graphs for Division and Multiplication

Evaluation of Certain Arithmetic Expressions

A set of elementary units can be conveniently applied in evaluating certain expressions, such as multiple products or sums, inner products, in O(m) recursive steps.[1] For example, the multiple product $P_c = \prod_{i=1}^{p} c_i$ can be easily evaluated as a degenerate case of a polynomial. Since

$$\left| \prod_{k=i}^{p} c_k - y_i^* \right| < r^{-m}, \quad i=1,2,\ldots,p$$

if $c_i = x$, $\forall i$, the E-method generates the positive integral powers of x : $x^2, x^3, x^4, \ldots, x^p$ in (m+1) steps on p elementary units. As before, it is assumed that factors c_i satisfy the range conditions. In general, any arithmetic expression which can be put into correspondence with a system L, can be evaluated in O(m) steps. For example, to evaluate

$$h = \frac{a(f+gc) + e(1+cd)}{1 + ab + cd}$$

one may solve

$$\begin{bmatrix} 1 & -a & 0 \\ b & 1 & -c \\ 0 & d & 1 \end{bmatrix} \underline{y} = \begin{bmatrix} e \\ f \\ g \end{bmatrix}$$

where $y_1 = h$.

Certain implications of the E-method on the solution of systems of linear equations and a possible reduction of the required complexity of iterative algorithms in general is discussed in some detail in [1]. No attempt is made here to discuss the implications the E-method may have in fields like digital signal processing or linear control systems. It may be expected that the proposed computational technique can be efficiently used in many special purpose computing systems or devices.

Conclusions

In this paper a recently discovered general evaluation method, amenable to an efficient hardware-level implementation, is presented. The proposed evaluation method, referred to as the E-method, is characterized by several important performance features and appears applicable in many common computational problems, such as the evaluation of polynomials and rational functions. It also brings together the following issues which, we believe, are of fundamental importance in the design of algorithms:

i) the choice of algorithmic representation compatible with the implementation environment;

ii) the problem of redundancy on the algorithmic level;

iii) the problem of redundancy in a number representation system.

The first issue is concerned with the problem of minimizing the number of algorithms to be implemented in order to solve a set of different problems. As is demonstrated here, the replacement of a given set of problems by a unique, isomorphic problem gives rise to a single algorithm to be implemented. The algorithm can, hopefully, satisfy the speed and cost objectives, among the other properties. And this, in general, would imply a small number of different primitive operators, simple enough to be efficiently implementable. In the E-method addition appears as the only required primitive operator. The corresponding algorithmic representation, considered as a way in which the computations are to be performed, has direct implications on the available parallelism and hence, the achievable speed. In a traditional approach, with four basic arithmetics as the primitive, indivisible operators, the parallelism is exposed or introduced by transformations of the original computation sequence so that the time dependencies between the required operations are minimized. The E-method demonstrates another approach in parallelism exposition: a systematic left-to-right, digit-wise processing minimizes the necessary delay between dependent computations and can achieve a parallelism up to one digit delay, having important properties like a simple, deterministic control. Furthermore, this approach is related to the second issue. Namely, it can effectively reduce the required complexity of an iteratively defined algorithm, by reducing the number of necessary operations. It also affects the choice of the primitive operators. It is of interest to remark that in many parallel algorithms it is, on the contrary, necessary to introduce redundant operations in order to achieve parallelism.[19]

The problem of redundancy in number representation systems has long been recognized as a central issue in achieving efficiently fast algorithms[20,21] and it will suffice here just to note that the E-method is another example where the redundancy in number representation has an essential role.

The theoretical basis of the E-method is simple yet, we believe, extendable to other interesting applications. Certainly, Algorithm E itself can be considered as a primitive operator which can be utilized in fast parallel schemes, not necessarily of the type defined by the E-method. The E-method can be incorporated in a computing system in two obvious ways: as an autonomous arithmetic processor with several elementary units, or, by providing the processors in a multiprocessor system with the capabilities of an elementary unit. Finally, it would be of interest to consider the changes in an instruction set which could make the E-method efficient for use on a software level.

Acknowledgement

The author is grateful to Professor James E. Robertson of the University of Illinois for the invaluable advice, encouragement and support and to Professors D.J. Kuck, A.H. Sameh and C.L. Liu, also of the University of Illinois, for their interest and advice.

The support of the Computer Science Department

of the School of Engineering and Applied Science of the University of California at Los Angeles during the preparation of this paper is sincerely appreciated.

References

1. Ercegovac, M.D., "A General Method for Evaluation of Functions and Computations in a Digital Computer," Ph.D. Dissertation, Department of Computer Science, University of Illinois at Urbana-Champaign, July, 1975.

2. Hart, J.F., Computer Approximations, New York, John Wiley & Sons, Inc., 1968.

3. Volder, J.E., "The CORDIC Trigonometric Computing Technique," IEEE Trans. Electron. Comput., Vol. EC-8, pp. 330-334, September, 1959.

4. Meggitt, J.E., "Pseudo Division and Pseudo Multiplication Processes," IBM J. Res. Develop,, Vol. 6, pp. 210-226, April,1962.

5. Specker, W.H., "A Class of Algorithms for lnx, expx, sinx, cosx, $\tan^{-1}x$, and $\cot^{-1}x$, " IEEE Trans. Electron. Comput.(Short Notes), Vol. EC-14, pp. 85-86, February, 1965.

6. Tung, C., "A Combinational Arithmetic Function Generation System," Ph.D. Dissertation, Department of Engineering, University of California, Los Angeles, California, June, 1968.

7. DeLugish, B.G., "A Class of Algorithms for Automatic Evaluation of Certain Elementary Functions in a Binary Computer," Ph.D. Dissertation, Department of Computer Science, University of Illinois, Urbana, Illinois, June, 1970.

8. Walther, J.S., "A Unified Algorithm for Elementary Functions," AFIPS Conference Proceedings, Spring Joint Computer Conference, pp. 379-385, 1971.

9. Chen, T.C., "Automatic Computation of Exponentials, Logarithms, Ratios and Square Roots," IBM J.Res. Develop., Vol. 16, pp. 380-8, July, 1972.

10. Franke, R., "An Analysis of Algorithms for Hardware Evaluation of Elementary Functions," Naval Post-graduate School, Monterey, California, AD-761519, May, 1973.

11. Campeau, J.O., "The Block-Oriented Computer," IEEE Trans. Comput., Vol. C-18, pp. 706-718, August, 1969.

12. Campeau, J.O., "Cellular Redundancy Brings New Life to an Old Algorithm," Electronics, pp. 98-104, July, 1969.

13. Campeau, J.O., "Communication and Sequential Problems in the Parallel Processor," in Parallel Processor Systems, Technologies and Applications, Edited by L.C. Hobbs, New York, Spartan Books, 1970.

14. Ho, I.T. and T.C. Chen, "Multiple Addition by Residue Threshold Functions and their Representation by Array Logic," IEEE Trans. Comput., Vol. C-22, pp. 762-767, August, 1973.

15. Kuck, D.J., "On the Speedup and Cost of Parallel Computation," Proceedings on the Complexity of Computational Problem Solving, The Australian National University, December, 1974.

16. Kung, H.T., "New Algorithms and Lower Bounds for the Parallel Evaluation of Certain Rational Expressions," Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania, Technical Report, February, 1974.

17. Svoboda, A., "An Algorithm for Division," Information Processing Machines, Vol. 9, pp.25-32, 1963.

18. Faddeev, D.K. and V.N. Faddeeva, Computational Methods of Linear Algebra, San Francisco, W.H. Freeman and Co., 1963.

19. Kuck, D.J., "Multioperation Machine Computational Complexity," Proceedings of Symposium on Complexity of Sequential and Parallel Numerical Algorithms, pp. 17-47, Academic Press, 1973.

20. Robertson, J.E., "A New Class of Digital Division Methods," IRE Trans. Electron. Comput., Vol. EC-7, pp. 218-222, September, 1958.

21. Avizienis, A., "Signed-Digit Number Representations for Fast Parallel Arithmetic," IRE Trans. Electron. Comput. Vol. EC-10, pp. 389-400, September, 1961.

Appendix

As a general example of the E-method we present the evaluation of $R_{3,4}(x)$ as an approximation to $\sinh(x)$, $x\varepsilon[0,1/8]$, with a precision of 13 decimal digits. The coefficients are from Hart[2] [SINH2002, p. 104, p. 216]:

$$p_0 = 0.0$$

$$p_1 = 0.5353890456087786*10^3$$

$$p_2 = 0.0$$

$$p_3 = 0.564627450687849*10^2$$

$$q_0 = 0.535389045608794*10^3$$

$$q_1 = 0.0$$

$$q_2 = -0.327694331123347*10^2$$

$$q_3 = 0.0$$

$$q_4 = 1.0$$

The other parameters are: $r=2$, $n=5$, $\zeta=3/4$, $\alpha=1/8$. For $x = 0.1019734533301$ the evaluation is illustrated in Figure 9. The generated value y_1^* satisfies

$$|(\sinh(x) - y_1^*)/\sinh(x)| < 2^{-45}.$$

J	$w_1^{(j)}$	d_1	d_2	d_3	d_4	d_5	y_1^*
1	0.000000000000000	0	1	0	0	0	0.000000000000000
2	0.203946906666018	0	0	0	0	0	0.000000000000000
3	0.467189381332036	0	0	0	0	0	0.000000000000000
4	0.815787662664072	1	0	0	1	0	0.125000000000000
5	-0.368424746711856	0	0	0	0	0	0.125000000000000
6	-0.736849493423712	-1	0	1	-1	0	0.093750000000000
7	0.526301013125756	1	0	-1	1	0	0.109375000000000
8	-0.947397673748468	-1	0	0	-1	0	0.101562500000000
9	0.105204052503036	0	0	0	1	0	0.101562500000000
10	0.210408105006408	0	1	1	0	0	0.101562500000000
11	0.624783111667234	1	0	-1	0	0	0.102539062500000
12	-0.750473766655532	-1	-1	1	0	0	0.102050781250000
13	0.295105540029181	0	1	0	0	-1	0.102050781250000
14	0.794158026711854	1	0	-1	0	0	0.102172851562500
15	-0.411683946562921	0	0	1	-1	1	0.102172851562500
16	-0.823367893125841	-1	1	-1	1	0	0.102142333984375
17	0.557211120406850	1	0	0	0	-1	0.102157592773437
18	-0.885577759163000	-1	-1	1	0	1	0.102149963378906
19	0.024897574973822	0	1	0	1	0	0.102149963378906
20	0.253742056860782	0	-1	-1	-1	0	0.102149963378906
21	0.303537206555464	0	0	0	0	0	0.102149963378906
22	0.607074413110923	1	0	1	1	0	0.102150440216064
23	-0.785851173778616	-1	1	0	-1	0	0.102150201797485
24	0.432244555812386	1	0	-1	1	-1	0.102150321006775
25	-0.735510888179228	-1	0	1	-1	1	0.102150261402130
26	0.528976234615444	1	-1	-1	0	0	0.102150291204452
27	-1.145990433829300	-1	1	1	0	0	0.102150276303290
28	-0.088033960998842	0	-1	0	1	0	0.102150276303290
29	-0.388014428657020	0	1	0	-1	-1	0.102150274440002
30	-0.556082750653860	-1	0	-1	0	1	0.102150274440002
31	0.887834498692280	1	-1	1	1	0	0.102150275371956
32	-0.428277909275620	0	0	0	0	0	0.102150275371956
33	-0.856555818551240	-1	0	-1	0	0	0.102150275139141
34	0.286888362897520	0	1	0	1	0	0.102150275139141
35	0.777723632455220	1	-1	0	-1	0	0.102150275197342
36	-0.644899641749740	-1	0	0	1	-1	0.102150275168243
37	0.703000716850520	1	1	1	-1	1	0.102150275182792
38	-0.390051660338780	0	0	0	0	-1	0.102150275182792
39	-0.781013320677560	-1	0	-1	0	1	0.102150275179153
40	0.439793358644880	0	1	0	0	0	0.102150275179153
41	1.083533623949960	1	-1	1	-1	0	0.102150275180062
42	-0.034879658760300	0	0	-1	0	0	0.102150275180062
43	-0.073759317520600	0	1	1	0	-1	0.102150275180062
44	0.056428271618980	0	0	-1	0	0	0.102150275180062
45	0.112856543237960	0	-1	1	1	0	0.102150275180062
46	0.021766617981574	0	1	0	0	1	0.102150275180062

Figure 9. Evaluation of $\sinh(x) \approx R_{3,4}(x)$