数理解析研究所講究録
第 673 巻 1988 年 62-79

Fast Automatic Differentiation and Interval Estimates of rounding errors

Koich KUBOTA[†] and Masao IRI [†]

1. Introduction

By means of Fast Automatic Differentiation [4,5,6], we can practically estimate rounding errors incurred in computed values of functions. Two kinds of estimates called "Absolute Estimates" and "Probabilistic Estimates" were proposed and it was shown by numerical experiments that they gave good approximations to the absolute values of rounding errors [6]. But we cannot theoretically rigorously assure those estimates to give a range for the function value containing the exact value (i.e., the value which would have been obtained without rounding errors). To obtain such a rigorous range for the function value, we may resort to a kind of interval arithmetic by replacing arithmetic operations to be executed in computation of the functions with the corresponding machine interval operations. However, when we do it for large-complicated functions, the width of the resulting interval may be terribly wide. In this paper, we propose an algorithm for calculating estimates which give rigorous and sharp upper bounds for the absolute values of rounding errors and which are proved to be "optimal" in the sense that will be defined later. We compare also the estimates obtained by means of the proposed algorithm with those obtained by machine interval operations and with the absolute estimates.

The proposed algorithm is a combination of Fast Automatic Differentiation and Machine Interval Operations. We assume that there are no errors in input data for the function and investigate the errors generated and accumulated in the course of computation only. (It is straightforward to extend to the case when the input data are contaminated with errors, too.) Our standpoint is as follows: "Each operation in computation is performed at the highest precision with the available machine so that we cannot know the exact value of rounding error or even its sign, but that we can only know the upper bound of the absolute value of the rounding error generated on the operation." This assumption will be accepted as a plausible one when rounding errors are rigorously discussed.

The basic principle in our algorithm is the "mean value theorem" in differential calculus. In interval analysis of a function with many variables, E. R. Hansen already used a method

[†] Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo, Bunkyo-ku, Tokyo 113, Japan.

based on the mean value theorem, but his method requires a computational time proportional to the product of the number of the variables and the computational time for the function itself [3,10]. Yu. V. Matiyasevich proposed another method which is an interval analysis with mean value theorem and which makes use of the idea of Fast Differentiation, and showed that the computational time of the method was independent of the number of variables [9]. Our method proposed in this paper extends that method of Matiyasevich's so as to be applied to rounding error estimation.

In section 2, we shall explain what kinds of functions we are to consider and make clear the concept of "computation with rounding errors". In section 3, we shall introduce the concept of machine interval operation, describe our algorithm and prove its optimality. In section 4, we shall show the results of numerical experiments with some observations.

2. Computation with rounding errors

2.1. Piecewise factorable functions and computational process

We assume that the operations used in the computation of a function are either unary or binary such as $+$, $-$, $\times$, $/$, exp, log, etc., which are continuously differentiable in their domain of definition. We call them *basic operations*. The functions we shall treat in this paper are the so-called *piecewise factorable functions* [7], of which the values are calculated by a finite sequence of basic operations represented in the form of a procedure or a program. The procedure for computing a piecewise factorable function may contain conditional branches and iterations which depend on the values of the input variables. We call the sequence of operations, which has actually been executed in the computation of the function value for the given values of the input variables, the *computational process*. A computational process consists of *computational steps*, each of which executes a basic operation and then stores the value in a variable called an *intermediate variable*. For a function with n variables $f(x_1, \ldots, x_n)$, its computational process is represented as shown in Figure 1. (Hereafter, descriptions will be only for binary operations; it should be understood that, for unary operations, the descriptions regarding the second argument will be deleted.) The number of intermediate variables in a computational process is equal to the number of the computational steps, r.

2.2. Floating-point systems and computations with rounding errors

On a computer, the result of a real operation is usually approximated by a value of the corresponding floating-point operation. For a specific computer (and a compiler), we call the set of floating-point numbers representable in it and the manner of rounding for real

6a

$$\begin{aligned} v_1 &\leftarrow \psi_1(u_{11}, u_{12}) \\ &\vdots \\ v_j &\leftarrow \psi_j(u_{j1}, u_{j2}) \\ &\vdots \\ (f =)\ v_r &\leftarrow \psi_r(u_{r1}, u_{r2}) \end{aligned}$$

$v_1, \ldots, v_r$:	intermediate variables;
$\psi_1, \ldots, \psi_r$:	basic operations;
u_{j1}, u_{j2}:	formal parameters of ψ_j. (Each of them corresponds to an input variable, a constant or an intermediate variable $v_1, \ldots, v_{j-1}$ $(j = 1, \ldots, r)$.)

Figure 1. Computational process

numbers used in it the *floating-point system.* The so-called *machine epsilon* ε_M expresses the supremum of the relative rounding errors occurring in the floating-point system.

When we perform computation in finite precision, i.e., with rounding errors, the computed values of intermediate variables as well as the computed value of the function are somewhat different from those values which we would have when we performed computation in infinite precision. It may happen that these differences in the computed values lead to different branches chosen at the conditional branches in the program and to division by a number with the different sign (which latter would imply the possibility of division by zero), and hence that the computational process realized by the computation in finite precision differs in structure from that realized by the computation in infinite precision. However, we shall assume in the following that the computational process remains the same in spite of the existence of rounding errors. (The validity of this assumption can be ascertained by means of interval analysis, as will be stated in section 3.)

We assume that values of input variables $x_1, \ldots, x_n$ are given. After the sequence of basic operations is executed giving rise to a computational process of $f$, the value of $f$ at $(x_1, \ldots, x_n)$ is obtained. When we compute the value of $f$ with a floating-point system, there arise rounding errors due to approximation. If the value of $f$ can be computed with interval arithmetic as we have assumed in the above, a unique structure of the computational process is determined. We denote the jth computational step actually performed with rounding errors by "$\bar{v}_j \leftarrow \bar{\psi}_j(\bar{u}_{j1}, \bar{u}_{j2})$" instead of "$v_j \leftarrow \psi_j(u_{j1}, u_{j2})$" in Figure 1 which would have been performed in the ideal case where no rounding error occurred. $\bar{\psi}_j$ indicates an basic operation with finite precision. (Over-lines will indicate "finite precision" in the following.)

$$\begin{array}{rcll}
\bar{v}_1 & \leftarrow & \bar{\psi}_1(\bar{u}_{11},\bar{u}_{12}) & = \psi_1(\bar{u}_{11},\bar{u}_{12}) + \delta_1 \\
 & \vdots & & \\
\bar{v}_j & \leftarrow & \bar{\psi}_j(\bar{u}_{j1},\bar{u}_{j2}) & = \psi_j(\bar{u}_{j1},\bar{u}_{j2}) + \delta_j \\
 & \vdots & & \\
(\bar{f} =)\ \bar{v}_r & \leftarrow & \bar{\psi}_r(\bar{u}_{r1},\bar{u}_{r2}) & = \psi_r(\bar{u}_{r1},\bar{u}_{r2}) + \delta_r
\end{array}$$

$\bar{v}_1,\ldots,\bar{v}_r$:	intermediate variables;
$\bar{\psi}_1,\ldots,\bar{\psi}_r$:	basic operations;
$\bar{u}_{j1},\bar{u}_{j2}$:	formal parameters of $\bar{\psi}_j$; (Each of them corresponds to an input variable, a constant or an intermediate variable $\bar{v}_1,\ldots,\bar{v}_{j-1}$ $(j = 1,\ldots,r)$.)
$\delta_1,\ldots,\delta_r$:	generated rounding errors.

Figure 2. Computational process with rounding errors

We define the *generated rounding error* δ_j associated with the jth computational step by

$$\bar{\psi}_j(\bar{u}_{j1},\bar{u}_{j2}) = \psi_j(\bar{u}_{j1},\bar{u}_{j2}) + \delta_j \tag{2.2.1}$$

(see Fig. 2).

Since actual parameters of ψ_j on the right-hand side of (2.2.1) are the computational results in finite precision, the generated rounding error δ_j is a local error which is "generated" in the execution of $\bar{\psi}_j$. By the assumption of the smoothness of basic operations, the difference between the jth computed result $\bar{v}_j$ and the exact value v_j is expressed as follows:

$$\begin{aligned}
\bar{v}_j - v_j &= \bar{\psi}_j(\bar{u}_{j1},\bar{u}_{j2}) - \psi_j(u_{j1},u_{j2}) \\
&= \psi_j(\bar{u}_{j1},\bar{u}_{j2}) - \psi_j(u_{j1},u_{j2}) + \delta_j \\
&= \frac{\partial \psi_j}{\partial u_{j1}}(u_{j1} + \theta_j \cdot (\bar{u}_{j1} - u_{j1}), u_{j2} + \theta_j \cdot (\bar{u}_{j2} - u_{j2})) \cdot (\bar{u}_{j1} - u_{j1}) \\
&\quad + \frac{\partial \psi_j}{\partial u_{j2}}(u_{j1} + \theta_j \cdot (\bar{u}_{j1} - u_{j1}), u_{j2} + \theta_j \cdot (\bar{u}_{j2} - u_{j2})) \cdot (\bar{u}_{j2} - u_{j2}) + \delta_j
\end{aligned} \tag{2.2.2}$$

with some θ_j between 0 and 1. $\partial\psi_j/\partial u_{j1}$ and $\partial\psi_j/\partial u_{j2}$ are the partial derivatives of the basic operation ψ_j, which we call *elementary partial derivatives* [4,5,6] and whose values at $(u_{j1} + \theta_j \cdot (\bar{u}_{j1} - u_{j1}),\ u_{j2} + \theta_j \cdot (\bar{u}_{j2} - u_{j2}))$ we denote by d_1^j and d_2^j, respectively:

$$d_i^j \equiv \frac{\partial \psi_j}{\partial u_{ji}}(u_{j1} + \theta_j \cdot (\bar{u}_{j1} - u_{j1}), u_{j2} + \theta_j \cdot (\bar{u}_{j2} - u_{j2})) \qquad (i = 1,2). \tag{2.2.3}$$

Thus, we get

$$\bar{v}_j - v_j = d_1^j \cdot (\bar{u}_{j1} - u_{j1}) + d_2^j \cdot (\bar{u}_{j2} - u_{j2}) + \delta_j. \tag{2.2.4}$$

The accumulated rounding error in the computed value of the function is equal to the difference between $\bar{v}_r$ $(= \bar{f})$ and v_r $(= f)$:

$$\bar{f} - f = \bar{v}_r - v_r = d_1^r \cdot (\bar{u}_{r1} - u_{r1}) + d_2^r \cdot (\bar{u}_{r2} - u_{r2}) + \delta_r. \tag{2.2.5}$$

In terms of computational graph [5,8], $G = (V, E, \partial^+, \partial^-, \omega, n, d)$, we can consider a set P_j of which elements are *paths* from the *vertex* v_j corresponding to the jth computational step to the *vertex* v_r corresponding to the value of the function. Each path, $p$, of P_j is a sequence of *edges* $e_1, \ldots, e_l$ such that the initial vertex of e_1 (denoted by $\partial^+ e_1$) is equal to v_j, $\partial^+ e_k$ is equal to the terminal vertex of $e_{k-1}$ (denoted by $\partial^- e_{k-1}$) $(k = 2, \ldots, l)$, and $\partial^- e_l$ is equal to v_r. Denoting by s_k the computational step number of $\partial^- e_k$, (i.e., $v_{s_k} = \partial^- e_k$) and by i_k the formal parameter number corresponding to e_k at $\partial^- e_k$ (i.e., $i_k = n(e_k)$), we consider a secuence of pairs $\{(s_k, i_k)\}_{k=1}^l$ corresponding to the path $p \in P_j$. Then, denoting by Q_j the set of such sequences corresponding to the paths which are the element of P_j for the jth computational step, thus, we can define

$$w_j = \sum_{\{(s_k,i_k)\}_{k=1}^l \in Q_j} d_{i_l}^{s_l} \cdot d_{i_{l-1}}^{s_{l-1}} \cdot \cdots \cdot d_{i_1}^{s_1} \tag{2.2.6}$$

$$\left(= \sum_{\{(s_k,i_k)\}_{k=1}^l \in Q_j} \frac{\partial \psi_r}{\partial u_{ri_1}} \cdot \frac{\partial \psi_{s_{l-1}}}{\partial u_{s_{l-1} i_{l-1}}} \cdot \cdots \cdot \frac{\partial \psi_{s_1}}{\partial u_{s_1 i_1}} \right)$$

$(j = 1, \ldots, r-1)$, and $w_r = 1$. Replacing $\bar{u}_{r1} - u_{r1}$ and $\bar{u}_{r2} - u_{r2}$ in (2.2.5) with the corresponding intermediate variables and substituting the expressions (2.2.4) repeatedly, we finally get

$$\bar{f} - f = \bar{v}_r - v_r = \sum_{j=1}^r w_j \cdot \delta_j, \tag{2.2.7}$$

where w_j is defined in (2.2.6) $(j = 1, \ldots, r)$. We shall discuss an algorithm for evaluating $|\bar{f} - f|$ rigorously in the following section.

When we set $\theta_1 = 0, \ldots, \theta_r = 0$, then we have

$$w_j = \frac{\partial f}{\partial v_j} \tag{2.2.8}$$

due to the chain rule of differentiation of a compound function. Thus, we have the linear approximation L_f:

$$\bar{v}_r - v_r \simeq L_f \equiv \sum_{j=1}^r \frac{\partial f}{\partial v_j} \cdot \delta_j, \tag{2.2.9}$$

which affords a base of ***absolute estimation*** and ***probabilistic estimation*** in [4, 5, 6]. Here, we can practically calculate all the $\partial f / \partial v_j$ $(j = 1, \ldots, r)$ with the method of *Fast Automatic*

Differentiation. Since $|\delta_j| \leq |v_j| \cdot \varepsilon_M$ (ε_M: machine epsilon) holds on almost all computers, the absolute estimation A_f may be defined [4,5,6,8] as

$$A_f \equiv \sum_{j=1}^{r} \left| \frac{\partial f}{\partial v_j} \right| \cdot |v_j| \cdot \varepsilon_M, \tag{2.2.10}$$

which gives a practical and good approximation to the upper bound of the absolute value of the rounding error but which is not a rigorous upper bound because eq. (2.2.9) is already an approximate formula.

3. An algorithm for a rigorous upper bound of the absolute value of the rounding error

We will show an algorithm for calculating an interval $S = [s^l, s^h]$ such that $\bar{f} - f \in S$. We can get the rigorous upper bound of the absolute value of the rounding error by $|\bar{f} - f| \leq |S|$ (where, for an interval $X \equiv [x^l, x^h]$, $|X|$ is defined to be $\max\{|x^l|, |x^h|\}$).

3.1. Interval operations and machine interval operations

By a *machine interval* we shall mean an interval of real numbers $[a^l, a^h]$ such that both a^l and a^h are floating-point numbers. We sometimes refer to an interval of real numbers simply as an *interval*. An *interval operation* is the operation which takes intervals as arguments and produces an image (which is also an interval) of the intervals of arguments by the corresponding real arithmetic operation. A *machine interval operation* is the operation which takes machine intervals as arguments and produces the narrowest possible machine interval that contains the result of the corresponding interval operation with the same arguments. Of course, machine intervals and machine interval operations depend on the floating-point system [1].

Substituting the corresponding interval machine operations for all the basic operations (and machine intervals as variables for all the variables) in the procedure for calculating a function f, we can construct a procedure which produces a machine interval $\bar{F}$ starting from the input intervals $[\bar{x}_1, \bar{x}_1], \ldots, [\bar{x}_n, \bar{x}_n]$ with width 0, where we set $\bar{x}_i = x_i$ for all i. $\bar{F}$ obviously contains both f and $\bar{f}$ so that the width of $\bar{F}$ is the rigorous upper bound for the absolute value of the rounding error. But it is known that the width of $\bar{F}$ may become too wide when the function f is computed with many computational steps, so that, in the next section, we will show a method to get a narrower upper bound for the absolute value of the rounding error than $\bar{F}$ after the computation of $\bar{F}$.

The history of the operations actually executed in the computation of f is the computational process in terms of machine interval operations. For a certain floating-point system and a set of input data, the "computability of $\bar{F}$ by machine interval operations" will mean that we can execute all the necessary machine interval operations without division by a machine interval including zero or comparison between two intersecting machine intervals. (If division by a machine interval including zero or comparison between intersecting machine intervals is required, we cannot proceed any further.) Thus, if $\bar{F}$ can be computed, the value of the function will certainly be computable under the assumption of section 2.2. (If $\bar{F}$ is not computable, the following arguments will be meaningless.) Evidently, if $\bar{F}$ is computable by a floating-point system represented by a machine epsilon ε_M, it is computable by any floating-point system of which parts for the mantissa and for the exponent are longer and which is represented by a smaller ε'_M, the resulting interval $\bar{F}'$ by the latter system being included in the interval $\bar{F}$ by the former.

3.2. Calculation of partial derivatives and estimation of rounding errors

Having computed $\bar{F}$ by machine interval operations, we shall proceed further as follows. If we denote by $\bar{V}_j$ the machine interval corresponding to an intermediate variable v_j in computing the machine interval $\bar{F}$ corresponding to f, we have

$$v_j, \bar{v}_j \in \bar{V}_j \qquad (j = 1, \ldots, r), \tag{3.2.1}$$

so that we have

$$u_{ji} + \theta_j \cdot (\bar{u}_{ji} - u_{ji}) \in \bar{U}_{ji} \qquad (j = 1, \ldots, r;\ i = 1, 2) \tag{3.2.2}$$

where we denote by $\bar{U}_{ji}$ the machine interval corresponding to u_{ji}.

Firstly, we substitute the machine interval operations for the operations in computation of elementary partial derivatives $\partial\psi_j/\partial u_{ji}$. Denoting by $\bar{D}^j_i$ $(j = 1, \ldots, r;\ i = 1, 2)$ the machine intervals calculated by those substituted machine interval operations, we have

$$\bar{D}^j_i \ni d^j_i = \frac{\partial \psi_j}{\partial u_{ji}}(u_{j1} + \theta_j \cdot (\bar{u}_{j1} - u_{j1}), u_{j2} + \theta_j \cdot (\bar{u}_{j2} - u_{j2})) \tag{3.2.3}$$

since $0 < \theta_j < 1$ $(j = 1, \ldots, r)$. (See Table 1, where $\oplus$, $\ominus$, $\otimes$, $\oslash$, etc. indicate the machine interval operation corresponding to $+$, $-$, $\times$, $/$, etc., respectively.)

Secondly, we calculate a machine interval $\bar{\Delta}_j$ containing the generated error δ_j which depends on the value of the intermediate variable, etc. For example, since it is the case in almost all computers that the absolute value of the generated error δ_j is less than $|\bar{v}_j| \cdot \varepsilon_M$, when computed value $\bar{v}_j$ is obtained, we may set $\bar{\Delta}_j \equiv [-\bar{\delta}_j, \bar{\delta}_j]$ with $\bar{\delta}_j \equiv |\bar{v}_j| \cdot \varepsilon_M$.[†] Thus, from eq. (2.2.5), we have

$$\bar{f} - f \in (\bar{D}^r_1 \otimes (\bar{u}_{r1} - u_{r1}) \oplus \bar{D}^r_2 \otimes (\bar{u}_{r2} - u_{r2})) \oplus \bar{\Delta}_r. \tag{3.2.4}$$

[†] Considering possible underflows, we may set $\bar{\delta}_j \equiv \max\{|\bar{v}_j| \cdot \varepsilon_M, pmin, -nmax\}$ where $pmin$ and $nmax$ are the minimum positive floating-point number and the maximum negative floating-point number, respectively, representable by the floating-point system used.

Table 1. Machine intervals corresponding to elementary partial derivatives

$\bar{D}_i^j$ corresponds to $\partial\psi_j/\partial u_{ji}$ where $v_j = \psi_j(u_{j1}, u_{j2})\Big|_{u_{j1} = v_k, u_{j2} = v_l}$.

ψ_j	$\bar{D}_1^j$	$\bar{D}_2^j$
$\pm$	$[1,1]$	$[\pm 1, \pm 1]$
$*$	$\bar{V}_l$	$\bar{V}_k$
$/$	$[1,1] \oslash \bar{V}_l$	$[-1,-1] \otimes \bar{V}_j \oslash \bar{V}_l$
exp	$\bar{V}_j$	—
log	$[1,1] \oslash \bar{V}_k$	—

Expanding $\bar{u}_{r1} - u_{r1}$, $\bar{u}_{r2} - u_{r2}$ in a similar manner repeatedly, we get the interval formula

$$\bar{f} - f \in ((\cdots(\bar{W}_r \otimes \bar{\Delta}_r) \oplus \cdots) \oplus \bar{W}_2 \otimes \bar{\Delta}_2) \oplus \bar{W}_1 \otimes \bar{\Delta}_1 \tag{3.2.5}$$

corresponding to (2.2.7). $\bar{W}_j$ $(j = 1, \dots, r)$ are the machine intervals which correspond to w_j in (2.2.6). They are computed by substituting the machine interval operations for the operations in the algorithm [4, 5, 6, 8] of Fast Automatic Differentiation as follows:

```
1) Initialization:—
    $\bar{W}_1, \dots, \bar{W}_{r-1} := [0,0]$;
    $\bar{W}_r := [1,1]$.
2) Computation:—
    for $j := r$ downto 1 do
        $a :=$ index of the intermediate variable of the first actual parameter of $\psi_j$;
        $b :=$ index of the intermediate variable of the second actual parameter of $\psi_j$;
        $\bar{W}_a := \bar{W}_a \oplus \bar{W}_j \otimes \bar{D}_1^j$;
        $\bar{W}_b := \bar{W}_b \oplus \bar{W}_j \otimes \bar{D}_2^j$.
```

(When ψ_j has only one argument or when an actual parameter of ψ_j is not an intermediate variable, a or b is defined. In such a case, we delete the part of computation for $\bar{W}_b$ or $\bar{W}_a$ from the above algorithm. Note that, if we can execute the machine interval operations for f to get $\bar{F}$, we can compute all $\bar{W}_j$ without division by a machine interval containing zero and without comparison of intersecting intervals. Furthermore, we may compute the machine interval product $\bar{W}_a \otimes \bar{D}_1^j$, etc. when they become necessary, instead of computing $\bar{D}_i^j$ in advance.) It is important to note that all the $\bar{W}_1, \dots, \bar{W}_r$ here can be computed in time proportional to r.

Thus, if we define A_F by

$$A_F \equiv \left| \bigoplus_{j=1}^{r} \bar{W}_j \otimes \bar{\Delta}_j \right|,$$

then $|\bar{f} - f| \leq A_F$ holds rigorously ($\bigoplus$ indicates summation by machine interval additions).

3.3. Algorithm for computing A_F and its computational complexity

The algorithm for A_F is summarized as follows:

(1) The machine interval operations are substituted for the real arithmetic operations appearing in the procedure for computing the value of f, and the machine interval variables for the real variables;

(2) The procedure constructed in (1) is executed with $\bar{X}_1 = [\bar{x}_1, \bar{x}_1], \ldots, \bar{X}_n = [\bar{x}_n, \bar{x}_n]$ as inputs, and then the computational process which have computed $\bar{V}_1, \ldots, \bar{V}_r (= \bar{F})$ is determined;

(3) After computing elementary partial derivatives in terms of intervals, $\bar{W}_1, \ldots, \bar{W}_r$ are computed by means of the algorithm of Fast Automatic Differentiation;

(4) Set $\bar{\Delta}_j = [-\bar{\delta}_j, \bar{\delta}_j]$, where $\bar{\delta}_j = |\bar{V}_j| \cdot \varepsilon_M \quad (j = 1, \ldots, r)$;

(5) Set $S = \bigoplus_{j=1}^{r} \bar{W}_j \otimes \bar{\Delta}_j$ ($\bigoplus$: machine interval summation);

(6) Set $A_F = |S|$.

The time complexity of each machine interval operation is equal (in order) to that of the ordinary real arithmetic operation corresponding to it. Therefore, (1) and (2) above are performed in time proportional to that for calculating the function alone. The computation of (3) is also performed in time proportional to that for calculating the function by means of Fast Automatic Differentiation. Hence, the time for computing A_F is proportional to that for calculating the function f alone. By a similar argument, the space required to compute A_F is also proportional to the time (not to the space) for calculating f.

3.4. Optimality property

Let us denote by $R = |\bar{f} - f|$ the absolute value of the rounding error in $\bar{f}$. Then, we have

$$R = \left| \sum_{j=1}^{r} w_j \cdot \delta_j \right| \tag{3.4.1}$$

(see (2.2.7)). Under the assumption in section 1, it may happen that all the signs of $w_1 \cdot \delta_1, \ldots, w_r \cdot \delta_r$ coincide with one another. In such a case, we have

$$R = \sum_{j=1}^{r} |w_j| \cdot |\delta_j|. \tag{3.4.2}$$

If we denote by $\bar{\delta}_j$ ($|\delta_j| \le \bar{\delta}_j$) the estimate which is a floating-point number and the tight upper bound for the absolute value of δ_j, we must anticipate the worst case where R is as large as

$$\bar{R} = \sum_{j=1}^{r} |w_j| \cdot \bar{\delta}_j. \tag{3.4.3}$$

Since A_F gives the width of a machine interval which is rigorously not smaller than $\bar{R}$, we have $\bar{R} \le A_F$.

Furthermore, under the assumption that

$$\partial f/\partial v_j \ne 0 \qquad (j = 1, \ldots, r), \tag{3.4.4}$$

we can prove that A_F is sharp enough, i.e., it is not too large compared with $\bar{R}$. More specifically, we shall prove that, as the machine epsilon approaches zero, the ratio $A_F/\bar{R}$ tends to 1.

Theorem. Let

$$A_F \equiv \left| \bigoplus_{j=1}^{r} \bar{W}_j \otimes \bar{\Delta}_j \right|, \qquad \bar{R} \equiv \sum_{j=1}^{r} |w_j| \cdot \bar{\delta}_j \quad \text{and} \qquad \bar{\Delta}_j \equiv [-\bar{\delta}_j, \bar{\delta}_j]. \tag{3.4.5}$$

Then, for arbitrary $\varepsilon > 0$, there exists $\eta > 0$ such that, if the computation with machine interval operations is performed with machine epsilon ε_M less than η, we have

$$1 \le \frac{A_F}{\bar{R}} \le 1 + \varepsilon. \tag{3.4.6}$$

□

Proof. The following (i) holds due to theorems[†] [1] for machine interval operations:

(i) For any $\varepsilon' > 0$ there exists an η_1 (> 0) such that, in a floating-point system with any machine epsilon ε_M less than η_1, w_j and $\bar{W}_j$ which we defined in §2.2 and §3.2 satisfy the relations:

$$0 \le |\bar{W}_j| - |w_j| \le |\bar{W}_j - w_j| \le \varepsilon' \qquad (j = 1, \ldots, r). \tag{3.4.7}$$

Moreover, due to the assumption (3.4.4), the continuous differentiability of f (in some domain) and the computability of $\bar{F}$, the following (ii) holds:

(ii) For $m \equiv \frac{1}{2} \min_j \left| \frac{\partial f}{\partial v_j} \right|$ (> 0), there exists η_2 such that w_j computed in a floating-point system with any machine epsilon ε_M less than η_2 satisfies

$$|w_j| > m \qquad (j = 1, \ldots, r).$$

[†] They are the theorem 4 and the theorem 5 in the chapter 4 of Alefeld and Herzberger's book [1].

From (i) and (ii), we have

(iii) For $\eta_3 \equiv \min\{\eta_1, \eta_2\}$,

$$1 \le \frac{|\bar{W}_j|}{|w_j|} \le 1 + \frac{\varepsilon'}{m}, \tag{3.4.8}$$

where w_j and $\bar{W}_j$ are computed in a floating-point system with machine epsilon ε_M less than η_3.

It is trivial that $A_F/\bar{R} \ge 1$ from the definition of A_F. Furthermore, it is seen that, even if we take account of the rounding errors generated during the calculation of the inner product with the machine interval operations at §3.3 (5), we have the inequality:

$$A_F \le \left(\sum_{j=1}^{r} |\bar{W}_j| \cdot \bar{\delta}_j\right) \cdot (1 + \varepsilon_M)^r. \tag{3.4.9}$$

In fact, the absolute value of the product $\bar{W}_j \otimes \bar{\delta}_j$ is bounded by $|\bar{W}_j| \cdot \bar{\delta}_j \cdot (1 + \varepsilon_M)$; for each addition in the inner product, the upper bound of the intermediate result is multiplied by $1 + \varepsilon_M$; and there are $r - 1$ additions in the inner product.

For any positive $\varepsilon (< 1)$, let $\varepsilon' = \varepsilon \cdot m/2$ and determine η_3 as in (iii). Then, if we compute the following expression in a floating-point system with machine epsilon ε_M less than $\eta_4 \equiv \min\{\eta_3, \varepsilon/4r\}$, we have

$$\begin{aligned}\frac{A_F}{\bar{R}} &\le \frac{\sum_{j=1}^{r} |\bar{W}_j| \cdot \bar{\delta}_j \cdot (1 + \varepsilon_M)^r}{\sum_{j=1}^{r} |w_j| \cdot \bar{\delta}_j} \\ &\le (1 + \frac{\varepsilon'}{m}) \cdot (1 + e^{r\varepsilon_M} \cdot r \cdot \varepsilon_M) \\ &\le 1 + \frac{\varepsilon'}{m} + 1.5 \cdot e^{1/4} \cdot \frac{1}{4} \cdot \varepsilon \\ &\le 1 + \varepsilon, \end{aligned} \tag{3.4.10}$$

where we made use of the inequalities $(1 + x)^r \le e^{rx} \le 1 + e^{rx} rx \quad (x \ge 0,\ r > 0)$. □

4. Numerical experiments

4.1. Simulation of machine interval operations

We have no special hardware nor software for direct execution of interval machine operations such as PASCAL-SC [2], so that we simulated machine interval operations on a VAX8600 machine with KCL[†] on ULTRIX[‡] as follows. The radix of the native floating-point system is binary, the mantissa of a number consists of 56 bits and the rounding is

[†] Kyoto Common Lisp

[‡] ULTRIX is an operating system.

toward the nearest. The machine epsilon is equal to 2^{-56}. On that machine, we implemented by a program binary floating-point systems which can have the mantissa of any length less than 56 bits. They are represented in terms of their machine epsilon, ε_M. The interval machine operations in a floating-point system represented by ε_M were performed as follows:

For each machine interval operation,

(1) we compute a machine interval $C \equiv [c^l, c^h]$ in the floating-point system represented by ε_M according to the definition of the interval operation;

(2) then, we calculate the machine interval $\bar{C} \equiv [\bar{c}^l, \bar{c}^h]$ that includes the perturbation of C by ε_M, i.e., such that

$$\bar{c}^l = \min\{c^l \cdot (1 - \varepsilon_M),\ c^l \cdot (1 + \varepsilon_M),\ c^h \cdot (1 - \varepsilon_M),\ c^h \cdot (1 + \varepsilon_M)\},$$
$$\bar{c}^h = \max\{c^l \cdot (1 - \varepsilon_M),\ c^l \cdot (1 + \varepsilon_M),\ c^h \cdot (1 - \varepsilon_M),\ c^h \cdot (1 + \varepsilon_M)\}.$$

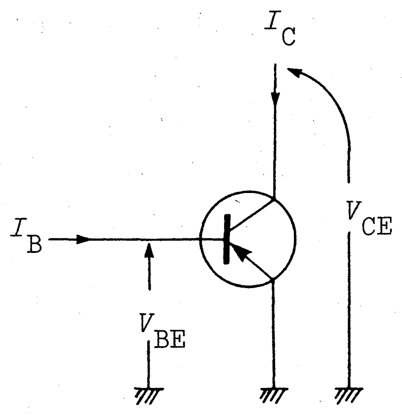
For simplicity, we neglected the effect of underflow.

4.2. Example 1: Ebers-Moll model of a transistor

We regarded the expression of the *base current* I_{B} in the Ebers-Moll model of a pnp-type transistor [4] (Fig. 3) as the definition of a function. The widths of the intervals given by the absolute estimate A_f, the estimate A_F proposed in this paper, the result with machine interval operations $\bar{F}$ are $2 \cdot A_f$, $2 \cdot A_F$, $d(\bar{F})$, respectively. They are compared for four different machine epsilons (ε_M in §4.1) 2^{-12}, 2^{-24}, 2^{-36} and 2^{-48} as shown in Figure 4. There it is seen that $2 \cdot A_f$, $2 \cdot A_F$ and $d(\bar{F})$ are almost equal to one another for such a small-scale function and that there is apparently no merit to compute A_F after computing $\bar{F}$.

4.3. Example 2: Solutions of linear systems by means of LU decomposition

A program for solving a linear system $A\boldsymbol{x} = \boldsymbol{b}$ for $\boldsymbol{x}$ with the matrix A and the vector $\boldsymbol{b}$ as input data by means of LU decomposition was regarded as the program for computing $\boldsymbol{x}$ as the functions with variables A and $\boldsymbol{b}$. Specifically, we regarded the first component x_1 of $\boldsymbol{x}$ as the value of a function with $n * (n+1)$ variables, $f(A, \boldsymbol{b})$ (n: the dimension of $\boldsymbol{x}$). We prepared ten pairs of a 5×5 matrix and a 5-dimensional vector $((A_1, \boldsymbol{b}_1), \ldots, (A_{10}, \boldsymbol{b}_{10}))$ the components of which were independently sampled from the uniform distribution on [-1,1]. We calculated A_f, A_F and $d(\bar{F})$ for the estimates of the rounding errors incurred in the values of $x_{1i} = f(A_i, \boldsymbol{b}_i)$ for the four machine epsilons mentioned in §4.1 ($i = 1, \ldots, 10$). There were cases where we could not compute $\bar{F}$ (hence, we could not compute A_F), because of large rounding errors for large machine epsilons. Therefore, we chose the absolute estimate of linear approximation, A_f, as the basis for comparison. Figure 5 shows A_F/A_f and $d(\bar{F})/(2 \cdot A_f)$. In



$$
\begin{aligned}
I_B &= -(1-\alpha_F)\cdot I_{ES}\cdot[\exp(-q\cdot V_{BE}/k\cdot T)-1] \\
&\quad -(1-\alpha_R)\cdot I_{CS}\cdot[\exp(q\cdot(V_{CE}-V_{BE})/k\cdot T)-1] \\
I_C &= -\alpha_F\cdot I_{ES}\cdot[\exp(-q\cdot V_{BE}/k\cdot T)-1] \\
&\quad +I_{CS}\cdot[\exp(q\cdot(V_{CE}-V_{BE})/k\cdot T)-1]
\end{aligned}
$$

Figure 3. Ebers-Moll model for a pnp-transistor

I_B, I_C:	base current and collector current
I_{ES}, I_{CS}:	saturation currents for the emitter-base junction and for the collector-base junction
α_F, α_R:	current transfer ratios
V_{BE}, V_{CE}:	voltages with the emitter as the datum node
T:	temperature
q:	electric charge of an electron
k:	Boltzmann constant

that figure, those points of which the ordinates are positioned at symbol "∞" indicate that the calculations corresponding to these points were interrupted by occurrence of division by an interval containing zero. Furthermore, we carried out similar experiments for 10-dimensional matrices and vectors (Fig. 6). In Table 2, part of the results of the latter experiments (10-dimensional case) is shown numerically, where it is seen that the number of the significant digits for the intervals guaranteed by A_F is remarkably greater than that by $\bar{F}$.

It may be observed that, in most cases, the widths of machine intervals obtained by the naïve machine interval operations are $10^1 \sim 10^5$ times larger for 5-dimensional linear systems, and $10^4 \sim 10^8$ times larger for 10-dimensional linear systems, respectively, than those obtained by the method proposed in this paper. In summary, it can be said that our estimate A_F not only gives a rigorous upper bound but also a sharp upper bound of the absolute value of the rounding error. In usual situations (where ε_M is less than 2^{-24} for 5-dimensional linear systems, and less than 2^{-48} for 10-dimensional linear systems), it is also observed that A_f is a good practical approximation of the upper bound of the absolute value of the rounding error.

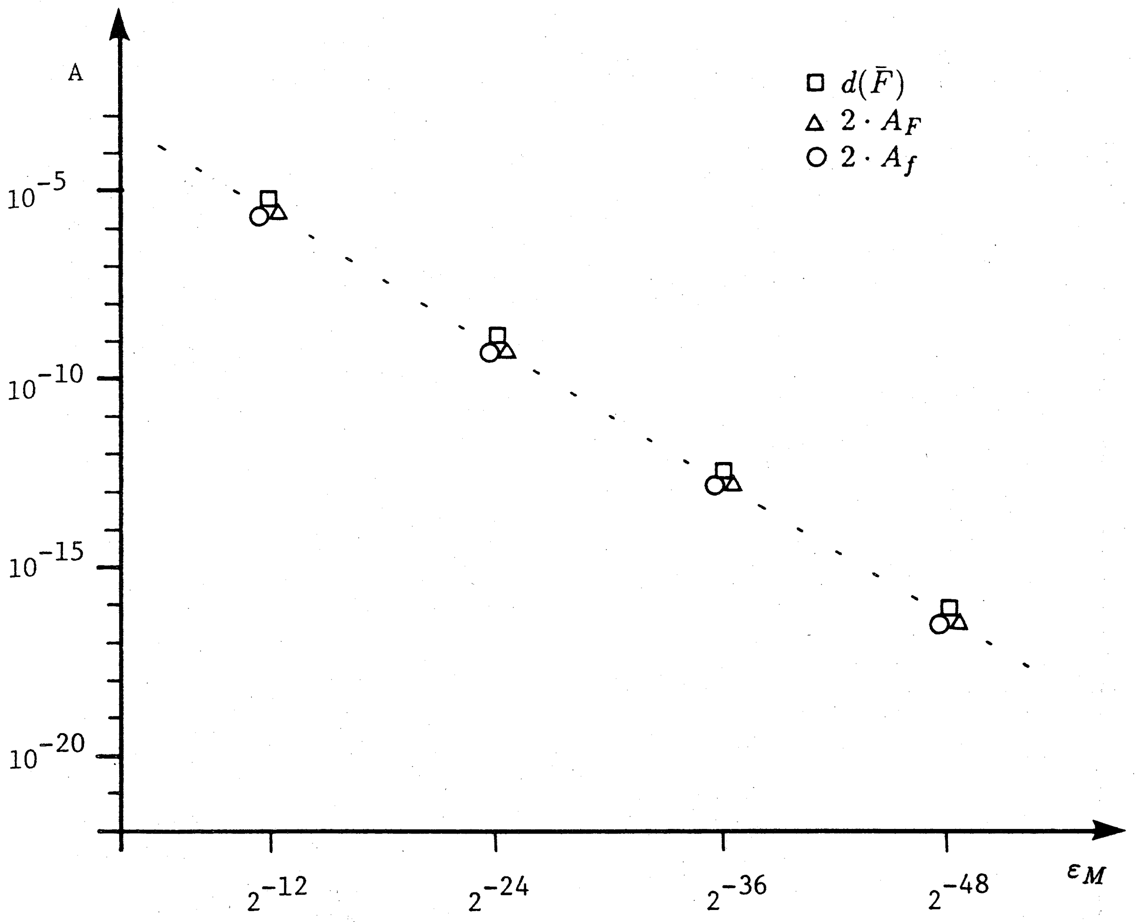


Figure 4. Comparison of the widths of guaranteed intervals containing the exact value for the function which is the base current in the Ebers-Moll model

$I_B \simeq -1.04 \times 10^{-4}$A
where $V_{BE} = -0.4$V, $V_{CE} = -1.0$V, $I_{ES} = 1.0 \times 10^{-9}$A,
$I_{CS} = 2.0 \times 10^{-9}$A, $\alpha_F = 0.98$, $\alpha_R = 0.5$, $T = 300$K,
$q = 1.602 \times 10^{-19}$C, k $= 1.38066 \times 10^{-23}$J/K

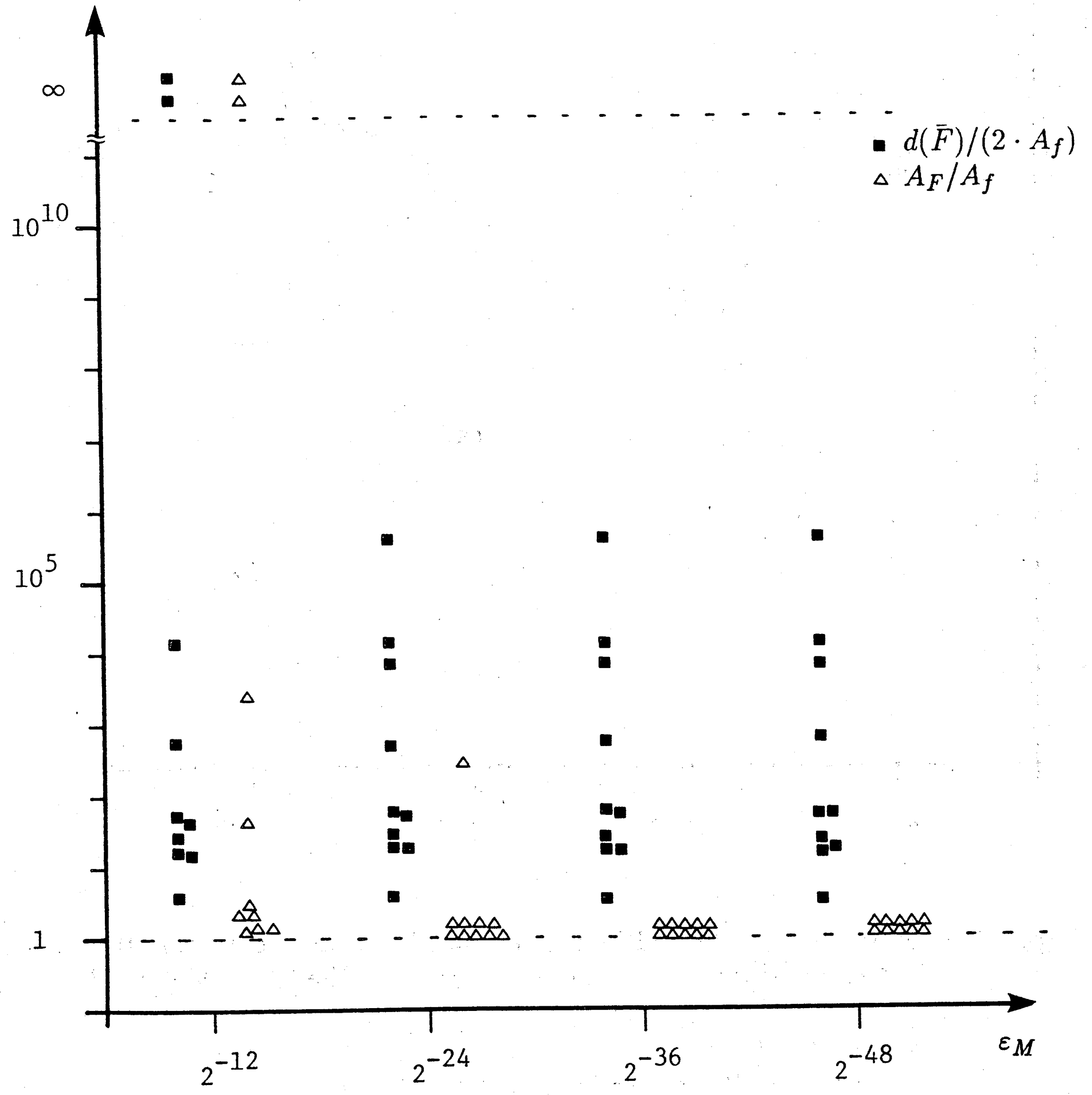


Figure 5. Comparison of the widths of guaranteed intervals for 5-dimensional linear systems

Points of which ordinates are positioned at symbol "∞" indicate that the calculations corresponding to those points were interrupted by occurrence of division by an interval containing zero.

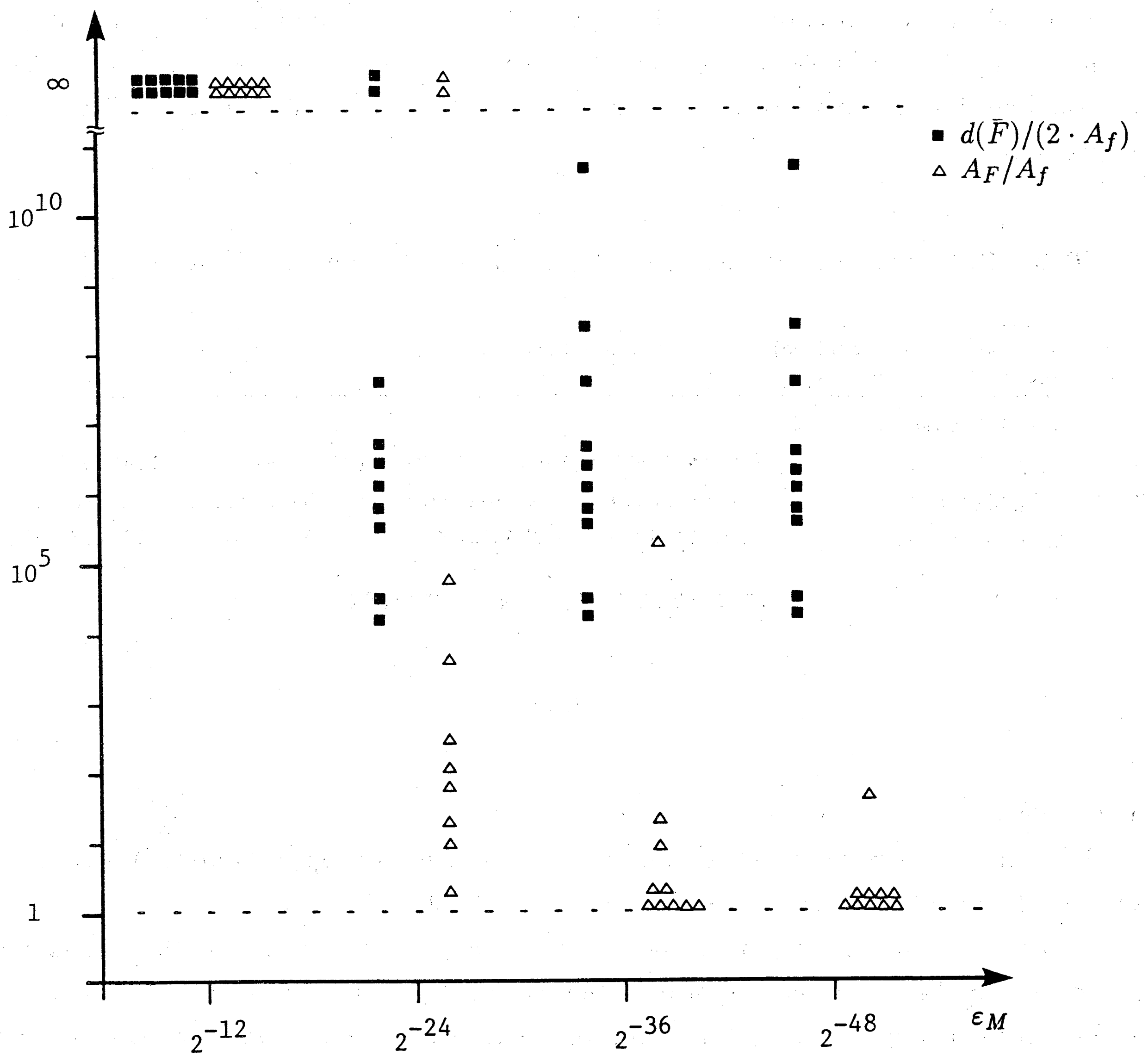


Figure 6. Comparison of the widths of guaranteed intervals for 10-dimensional linear systems

Points of which ordinates are positioned at symbol "∞" indicate that the calculations corresponding to those points were interrupted by occurrence of division by an interval containing zero.

Although sometimes hardware and software [2] which compute inner products without rounding errors are available, we considered here a situation where each of the multiplications and the additions in computing the inner product of vectors generates a rounding error individually.

Table 2. Guaranteed intervals with $\bar{F}$ and A_F for a 10-dimensional linear system

ε_M	computed value and significant digits (underlined) estimated by means of A_f	guaranteed interval with $\bar{F}$	guaranteed interval with A_F
2^{-12}	$\underline{\texttt{0}}\texttt{.7438964843}$	—	—
2^{-24}	$\underline{\texttt{0.7542}}\texttt{193532}$	[-3344.4582519,3346.1162109]	[0.4637820125,1.0446566939]
2^{-36}	$\underline{\texttt{0.7542197}}\texttt{853}$	[0.0575954438,1.4508441332]	[0.7542197555,0.7542198151]
2^{-48}	$\underline{\texttt{0.7542197855}}$	[0.7540463030,0.7543932679]	[0.7542197855,0.7542197855]†

†[0.754219785475757,0.7542197854840781]

5. Conclusion

We proposed a practicable algorithm for a rigorous and sharp upper bound of the absolute value of the rounding error incurred in the computed value of a function.

Through the numerical experiments we have done it is observed that the estimate of rounding error based on linear approximation is usually good enough, but it is important at least theoretically to establish a technology with which we can get rigorous and sharp estimate.

References

[1] G. Alefeld and J. Herzberger: *Introduction to Interval Computations.* Academic Press, New York, 1983.

[2] G. Bohelender, C. Ullrich, J. W. Gudenberg, and L. B. Rall: *Pascal-SC — A Computer Language for Scientific Computation.* Academic Press, Orlando, 1987.

[3] E. R. Hansen: A Generalized Interval Arithmetic. K. Nickel (ed.): *Interval Mathematics*, Lecture Notes in Computer Science 29, Springer-Verlag, Berlin, 1975, pp. 7–18.

[4] M. Iri: Simultaneous Computation of Functions, Partial Derivatives and Estimates of Rounding Errors — Complexity and Practicality. *Japan Journal of Applied Mathematics*, Vol. 1, No. 2 (1984), pp. 223–252.

[5] M. Iri and K. Kubota: Methods of Fast Automatic Differentiation and Applications. *Research Memorandum RMI 87-02,* Department of Mathematical Engineering and Information Physics, University of Tokyo, 1987.

[6] M. Iri, T. Tsuchiya, and M. Hoshi: Automatic Computation of Partial Derivatives and Rounding Error Estimates with Applications to Large-scale Systems of Nonlinear Equations. To appear in *Journal of Computational and Applied Mathematics*, 1988.

[7] G. Kedem: Automatic Differentiation of Computer Programs. *ACM Transactions on Mathematical Software*, Vol. 6, No. 2 (1980), pp. 150–165.

[8] K. Kubota and M. Iri: Formulation and Analysis of Computational Complexity of Fast Automatic Differentiation (in Japanese). *Transactions of Information Processing Society of Japan*, Vol. 29, No. 6 (1988), pp. 551–560.

[9] Yu. V. Matiyasevich: Veshchestvennye Chisla i ÈVM. *Kibernetika i Vychislitel'naya Tekhnika*, Vypusk 2 (1986), pp. 104–133.

[10] L. B. Rall: Improved Bounds for Ranges of Functions. K. Nickel (ed.): *Interval Mathematics 1985*, Lecture Notes in Computer Science 212, Springer-Verlag, Berlin, 1985, pp. 143–155.