資　　料

# 誤差評価の可能な多重精度演算*

大　中　幸三郎*　安　井　　裕**

**Abstract**

It is a very difficult problem to estimate the error bounds in every algorithm. But if calculating error is determined in each operation, it is very useful matter for error estimation. In this paper, we give the multiple-precision arithmetic with error estimation. Our methods of error estimation are round-off control, error arithmetic and interval arithmetic. The multiple-precision arithmetic packeges can be used in variable length during computation. We show the details of algorithm and some results.

## 1. まえがき

数値計算によって得られた解の確度の評価は，自動計算機による数値解析上，重要な事柄である．高速に処理するための工夫も興味深い内容をもつものではあるが，数値解析上の重要な関心は，打切り誤差，丸め誤差，入力誤差の伝播などの決定にもある．しかしながら，どのような計算についても意味のある計算の過程として，その誤差の振舞いを知ろうとするときには，計算過程の個々について，何らかの形で直接的に誤差の影響をつかんだ演算を行う必要がある．このような演算誤差の評価に対する手段としては，丸めの制御，significance arithmetic[1),2)], interval arithmetic[3),4)] などの方法がある．しかしながら，これらの例はすべて単精度の場合に限定されており，また，単なる多重精度演算であれば多くの発表[5)]があるが，いずれも積極的に誤差評価のための機能をもっているものはない．これらの significance もしくは interval arithmetic などの手段をもちいて演算誤差の評価を行うとき，特に ill-condition の問題に適用する際には，単精度，倍精度では不充分となることが予想される．したがって，誤差評価機能をもつ多重精度演算プログラムは，実用上，大いに価値があるものと考える．ここに，誤差評価機能をもつ多重精度演算における問題点と適用例について述べる．プログラムは互換性と客観性を考えて FORTRAN* で作成しているので，さらに他の言語で作成することも容易であろう．

## 2. 数の丸めの定義とその取扱い

現在，計算機の hardware における数の丸めは，おおむね切り捨てまたは四捨五入であり，softwareである言語のコンパイラによって作られる目的プログラムの丸めの制御は自由になっていない．いままでに作られた多重精度演算プログラムでも同様であることが多い．次にわれわれが行なっている丸めの定義をする．$x$ を $p>x\geqq 1$ に正規化を行った $p$ 進小数とし，(1)式で示す．

$$x=x_0.x_1x_2\cdots x_m. \tag{1}$$

ただし $\begin{cases} x_0=1,\ 2,\ \cdots,\ p-1, \\ x_i=0,\ 1,\ \cdots,\ p-1 \end{cases}$　$i=1,\ 2,\cdots,m.$

丸めは有効桁を短縮するのであるから，正規化を行っても一般性は失われない．（1）式を小数点以下 $q-1$ 桁($q\leqq m$) に丸める場合を考える．ここで $[x]_r{}^q$ を次式で定義し，この式で丸めの定義とする．

$$[x]_r{}^q\equiv\begin{cases} x_0.x_1x_2\cdots x_{q-1} & x_q<r. \\ x_0.x_1x_2\cdots x_{q-1}+p^{-(q-1)} & x_q\geqq r. \end{cases} \tag{2}$$

ただし　$r=0,\ 1,\ \cdots,\ p.$

切り上げ以外の丸めは　$x_q$　だけに注目すればよいが，切り上げの場合は $x_q$，$x_{q+1}$，… と無限に調べる

* Multiple-precision Arithmetic with Error Estimation, by Kohzaburo OHNAKA and Hiroshi YASUI (Department of Applied Physics, Faculty of Engineering, OSAKA University)

** 大阪大学工学部応用物理学科

* DFLOAT をもちいた以外は JIS FORTRAN（水準 7,000）に含まれる．

必要があるので，(2)式で $r=0$ としたときを切り上げとした．$r=0$ のときと無限桁調べる切り上げ（真の切り上げ）の差を示すために，丸めた結果が $x_0.x_1x_2\cdots x_{q-1}$ になる $x$ の区間を次に示す．

$r=0$

$$x_0.x_1\cdots x_{q-1}-p^{-(q-1)}\leqq x<x_0.x_1\cdots x_{q-1}. \quad (3)$$

真の切り上げ

$$x_0.x_1\cdots x_{q-1}-p^{-(q-1)}<x\leqq x_0.x_1\cdots x_{q-1}. \quad (4)$$

(3)，(4) 式では等号に差があるが，(3)式の方が(4)式よりも丸め誤差が大きく，誤差評価上，妥当性があることと，真の切り上げを行うには丸めの手続きを全く別に作る必要があるので，$r=0$ を切り上げとした．

## 3. 誤差評価の可能な多重精度演算における基本的演算機能

われわれは以下に述べる理由から，実際のプログラムの作成に際し，次の4種類の機能をもつパッケージを作った．

実数演算における誤差評価に必要な機能として，丸めの制御を最小限もたねばならない．そのため，丸めの制御を可能にするとともに，利用に際して基本的でもっとも，自由度の高い記述のできるパッケージとして，第1 operand を accumulator 的に考えた演算 X←X⊛Y を行うのが M 型であり，作業用レジスタとしての配列や数としての配列の長さまで引数に指定することができる．したがって，計算中の各演算毎に精度を変えながら計算することも他の型にくらべて容易であり，**4.** で述べるように演算時間の短縮などに機能を発揮できる．M 型は自由度が高いが引数が多いので，さらにマクロ的にもちいられるようにしたのがS型である．これは引数としては operand の3つだけであり，operand X，Y，Z に対して演算 Z←X⊛Y が行われる．また，記憶場所の節約も行なっている．M，S 型による演算誤差の評価の手段としては主として丸めの制御を利用することになる．

一般に行われている演算誤差の評価の手法として，同一のアルゴリズムを単精度，倍精度の両方で演算した結果について比較し，一致した桁までを解として採用する方法がしばしば行なわれるが，この方法は必ずしもすべての場合に妥当性はない．特に ill-condition の場合には無力となるので，このような場合においても演算誤差の評価を行うために，誤差を独立した値としてもち，その誤差の演算をともなった演算方式を行う E 型をもっている．しかしながら，この場合にも **8.** で述べるような評価の範囲にとどまるので，さらにアルゴリズムそのものの誤差を含めて誤差評価を行うことのできる interval arithmetic を行うI型をもっている．これは解が必ず存在する区間を決定できる．

配列名 *MX*

| | | | |
|---|---|---|---|
| (1) | 1 | 符号部 | $\{MX(1)\}=\begin{cases}1 & \text{正数}\\ 0 & 0\\ -1 & \text{負数}\end{cases}$ |
| (2) | 0 | 指定部 | |
| (3) | 3 | 整数部 | |
| (4) | 14159 | 小数部 | |
| (5) | 26535 | ⋮ | 仮数部 |
| (6) | 89793 | | |
| (7) | 23846 | | $\{MX\}=3.14159\cdots83280\times t^0$ |
| (8) | 26433 | | ただし |
| (9) | 83280 | | $t=10^n$（この例では $n=5$） |

Fig. 1 Format of M type

ここに述べた誤差評価の手段以外に significance arithmetic をもちいる方法もあるが，これは E 型の機能に含まれているものと考えている．

さて，実際の場合には1つの多重精度実数は一次元整数配列で表わし，たとえば $\pi$ を 10 進 31 桁で示すと図1となる．仮数部の配列要素に 10 進で $n$ 桁を入れるときの $n$ の条件は，$10^{2n}$ が単精度整数型であふれないことである．これは各要素毎の乗算で演算結果があふれないためであり，図1は $n=5$ の場合である．指数部は $10^n$ 毎の指数を示すように定義する．したがって，図1で $\{MX(2)\}^*=1$ なら $\{MX\}=314159.26\cdots$ を示す．

## 4. 演算中における精度の変更とその利点

われわれの多重精度演算ルーチンは多重精度の数を表わす配列の長さをパラメータとしているので，演算中に各 operand の仮数部の長さを変化させて演算時間を短縮することができる．その例としては反復法で解を求める場合，Taylor 展開で関数値を求める場合などがある．ここでは $y=x^2-a$ に Newton 法を適用して $\sqrt{a}$ を求める場合を示す．この場合の反復式は，

$$x_{n+1}=x_n-\frac{x_n^2-a}{2x_n}=\frac{x_n^2+a}{2x_n} \quad (5)$$

となる．(5)式を変形すると (6) 式が得られる．

$$\frac{x_{n+1}}{\sqrt{a}}-1=\frac{\sqrt{a}}{2x_n}\left(\frac{x_n}{\sqrt{a}}-1\right)^2. \quad (6)$$

$x_n$ が 10 進 $p$ 桁，$x_{n+1}$ が 10 進 $q$ 桁の精度で

* {name} と書くことにより name の内容を表わすものとする．

$\sqrt{a}$ に等しいとすれば,

$$\left|\frac{x_n}{\sqrt{a}}-1\right|\fallingdotseq 10^{-p},\quad \left|\frac{x_{n+1}}{\sqrt{a}}-1\right|\fallingdotseq 10^{-q} \qquad (7)$$

となる. (6)式の右辺が正であることに注目し, (7)式を (6) 式に代入する.

$$10^{-q}\fallingdotseq\frac{\sqrt{a}}{2x_n}10^{-2p}.$$

$n$ が充分大きいと $\sqrt{a}/x_n\fallingdotseq 1$ となるので上式を $q$ について解けば次式となる.

$$q\fallingdotseq 2p+\log_{10}2\fallingdotseq 2p+0.3. \qquad (8)$$

したがって, $x_n$ が 10 進 $p$ 桁の精度をもっていれば $x_{n+1}$ は近似的に $(2p+0.3)$ 桁の精度をもつ. ところが (8) 式の導出過程は無限桁演算を行う場合である. 無限桁演算を有限桁化して多重精度演算を行うので, 有限桁演算における精度の低下を考えて(8)式の 0.3 の項を無視すると, 1回反復する毎に $x_n$ の有効数字はおよそ倍となる. 多重精度演算の場合は初期値 $x_0$ を倍精度実数型演算で求められ, $\sqrt{a}/x_n\fallingdotseq 1$ を満足する. したがって $x_0$ の精度を $p$ 桁とすれば $x_n$ の精度はほぼ $2^n p$ 桁となる. すなわち, $x_n$ を求める反復では $2^n p$ 桁よりも少し長い程度で演算を行えばよく, 必要以上に長い桁数で演算をしても演算時間を費すだけである. この考え方のもう一つの利点はあらかじめ反復回数を定められ, 反復毎に停止則を適用しなくてよい点である. 多重精度演算での大小判定は長い配列要素毎に1つ1つ調べる必要があり, 演算時間の点で不利なので, 停止則による判定を省略できるのは非常に有利である.

## 5. 除算ルーチンとその加速法

多重精度演算では除算が最も複雑で時間がかかる. 除算の演算方法については多くの論文[6)~8)]があるが, われわれは商の精度の確認と, 演算時間の点で有利な divide and correct 法を基本として加速を考える. この方法では, 各配列要素毎に推定値を求めて修正するが, 除算は指数部と仮数部がおのおの独立して演算されるので, 仮数部のみに注目すればよい. 分母, 分子の仮数部を次のように定義する.

分子 $x=x_0.x_1\cdots x_k$,
$x_0=1, 2, \cdots, t-1,\ x_i=0, 1, \cdots, t-1.$

分母 $y=y_0.y_1\cdots y_k$,
$y_0=1, 2, \cdots, t-1,\ y_i=0, 1, \cdots, t-1.$
ただし $i=1, 2, \cdots, k,\quad t=10^n$

商 $z$ が小数点以下 $j$ 桁目まで求まったとする.

$z^j=z_0.z_1\cdots z_j$,
$z_i=0, 1, \cdots, t-1,\ i=0, 1, \cdots, j.$

このときの余り $r^j$ は次式で与えられる.

$$r^j=r_jr_{j+1}.r_{j+2}\cdots r_{j+k}\times t^{-(j+1)}\equiv R^j\times t^{-(j+1)}.$$

ただし $r_{j+i}=0, 1, \cdots, t-1,$
$i=0, 1, \cdots, k.$

次に $z_{j+1}$ を求めるのだが, その方法はすでに発表されたものについて調べると次の2つになる.

(A) 推定値を $z_{j+1}'=(r_j\times t+r_{j+1})/y_0$ とする.

(B) 第1近似値を $a^{(1)}=(r_j\times t+r_{j+1})/(y_0+1)$ とし, $R^j-a^{(1)}\times y$ をあらためて $R^j$ とおき $a^{(2)}=(r_j\times t+r_{j+1})/(y_0+1)$ として $z_{j+1}=a^{(1)}+a^{(2)}+\cdots$ とする.

この2つの方法はどちらも演算時間がかかる. その例として $n=5$ で $y$, $r^j$ が次の場合を考える.

$$\left.\begin{array}{l} y_0=1,\ y_1=50000,\ y_2=0,\ y_i=0 \\ r_j=0,\ r_{j+1}=r_{j+2}=99999,\ r_{j+i}=0, \\ \quad \text{ただし}\quad i=3, 4, \cdots, k. \end{array}\right\} \qquad (9)$$

このとき(A)では $z_{j+1}'=99999$ となり, 真値 $z_{j+1}=66666$ とくらべると 33333 回の修正ループをまわることになる. 一方 (B) では $a^{(9)}$ まで計算しなければならない. あらかじめ分母を正規化して (A), (B) をもちいる方法[8)]もあるが, 正規化のための処理が必要なので, 演算時間と誤差評価の点で好ましくない. 演算時間を短縮するには推定値の精度を上げればよいのであるが, 一般には $r_j\neq 0$ なので単精度整数型演算では精度の向上はできない. したがって, われわれは $y$ と $R^j$ を単精度実数化して $z_{j+1}'$ を求める. 単精度実数型演算を 10 進 10 桁とすれば (9) 式の例では $z_{j+1}'=99999.99999/1.500000000=66666.66666$ となる. $z_{j+1}$ は整数なので $z_{j+1}'$ の整数部をあらためて $z_{j+1}'$ とすれば $z_{j+1}'=66666$ となり, $z_{j+1}$ に等しくなる. このようにして求められた $z_{j+1}'$ は $|z_{j+1}-z_{j+1}'|\leqq 1$ と考えられ*, $z_{j+1}$ を求める修正ループ高々1回でもすみ, 加速され, 乗算と同じ程度の演算速度で処理することができる.

## 6. 誤差評価の可能な多重精度演算パッケージ

### 6.1 M型 (machine language like arithmetic)

数の表現は図1の基本的表現である. 演算誤差の評価においては桁落ちの程度の把握が重要なので, 加減

* この不等式が成立するように $n$ に条件をつける必要があるが, 通常の hardware では 3. で述べた条件でよい.

配列名
*LX*

| | | |
|---|---|---|
| (1) | | 1 |
| (2) | | 0 |
| (3) | | 3 |
| (4) | 1 4 1 5 9 2 6 5 3 5 | |
| (5) | 8 9 7 9 3 2 3 8 4 6 | |
| (6) | 2 6 4 3 3 8 3 2 8 0 | |

Fig. 2 Format of S type

算の場合に桁落ちを示す指標をつけた．この指標は桁落ちした配列要素数がセットされるので，大幅な桁落ちが起るときには有効である．M型で演算誤差を評価するには桁落ちの指標以外に丸めの制御，演算精度を変える方法などがある．

### 6.2 S型 (symbolic language like arithmetic)

図1では小数部一要素に 10 進で $n$ 桁を格納したが，$10^{2n}$ があふれないので記憶場所の節約のため，小数部に $2n$ 桁格納したのがS型であり，図2に示す．記憶場所の節約には符号部と整数部を1つにまとめてもよいが，演算中に分離を行う必要があることとM↔S変換の速度を早くするために符号部，指数部，整数部の定義は M 型と同一とした．

ここで重点をおいたのは演算時間の短縮のため，M↔S変換をさけることである．乗除算では M 型に展開して演算を実行する必要があるが，加減算では指数部の差が偶数のときは展開しなくてよく，この場合は配列一要素分の $2n$ 桁を operand とする加減算に分解できる．加算の場合のアルゴリズムを示そう．operand がおのおの $2n$ 桁の数であるときに和は $2n+1$ 桁となり，あふれるので，加算を次のように変形する．

$$c=(a-10^{2n})+b,\quad 0\leqq a<10^{2n},\quad 0\leqq b<10^{2n}.$$

この場合の結果は次の通りである．

$$\begin{cases} c\geqq 0 \text{ のとき} & \text{sum } c, \quad \text{carry } 1, \\ c<0 \text{ のとき} & \text{sum } c+10^{2n}, \quad \text{carry } 0. \end{cases}$$

S型の演算は Z←X⊛Y であるが，X，Y，Z は相異なる配列名でなくてもよい．桁落ちの指標，演算誤差の評価方法については M 型と同一である．

### 6.3 E型 (error arithmetic)

図1に誤差項を付加し，演算誤差の評価に役立つものとした．図3にその例を示す．第1要素に仮数部の長さを表わしたのは，誤差の伝播による有効桁数の変化のためである．第1要素の内容の定義から，0に対して誤差は考えず，$\{MX(1)\}=0$ なら真の0とする．

E型で最も困難なのは誤差項の計算である．

$$(x\pm\delta x)\circledcirc(y\pm\delta y)=z\pm\delta z,\ \delta x,\delta y,\delta z\geqq 0. \quad (10)$$

配列名
*MXE*

| | | |
|---|---|---|
| (1) | 7 | 符号＋仮数部の長さ |
| (2) | 1234567890 | 誤差項 最下位2要素に対する誤差) |
| (3) | 0 | $0\leqq\{MXE(2)\}\cdot 10^{2n}$ |
| (4) | 3 | |
| (5) | 14159 | $a$…誤差を含む部分 |
| (6) | 26535 | $b$…仮数部の長さ(この例では7) |
| (7) | 89793 | $b$ |
| (8) | 23846 | $\{MXE\}=3.14159\cdots 83280$ |
| (9) | 26433 | $\quad\times 1234567890\times t^{(-1)}\times t^{a}$ |
| (10) | 83280 | ただし $t=10^n$(この例では $n=5$) |

Fig. 3 Format of E type

除算の場合をのぞいて $\delta_z$ は次式で与えられる．

$$\delta z=\begin{cases}\delta x+\delta y & \text{加減算},\\ |x|\delta y+|y|\delta x+\delta x\delta y & \text{乗　算}.\end{cases} \quad (11)$$

除算の場合は次のようになる．

$$\delta z=\max\left(\left|\frac{x+\delta x}{y+\delta y}-\frac{x}{y}\right|,\ \left|\frac{x-\delta x}{y+\delta y}-\frac{x}{y}\right|,\right.$$
$$\left.\left|\frac{x+\delta x}{y-\delta y}-\frac{x}{y}\right|,\ \left|\frac{x-\delta x}{y-\delta y}-\frac{x}{y}\right|\right). \quad (12)$$

ただし，$(y-\delta y)(y+\delta y)>0$.

演算時間の点から (12) 式を次式で近似する．

$$\delta z=\frac{\delta x}{|y|}+\frac{\delta y}{|y|^2}\left(1+\frac{\delta y}{|y|}\right)(|x|+\delta_x). \quad (13)$$

(13) 式の導出過程で $1/(1-a)\fallingdotseq 1+a+a^2$ を使っているので，$a^3$ 以上の項を無視するために(仮数部の長さ)≧4 とする．E 型では誤差は $\{MX(2)\}$ と $|\{MX(1)\}|$ の2つの整数で表現される．(11)式は整数型演算でも処理できるが (13) 式に除算がある関係上，$\{MX(2)\}$ を倍精度実数化して (11)，(13) 式を評価する．誤差評価の場合は E 型の数は次の4つの部分にわけられる．

$xp_i$, $yp_i$, $zp_i$…operand $x$, $y$, $z$ の指数部.
$xm_d$, $ym_d$, $zm_d$…$x$, $y$, $z$ の仮数部.
$xe_d$, $ye_d$, $ze_d$…$x$, $y$, $z$ の誤差項.
$xl_i$, $yl_i$, $zl_i$…$x$, $y$, $z$ の仮数部の長さ.

ただし，添字 $\begin{cases} i\cdots\text{単精度整数型},\\ d\cdots\text{倍精度実数型}.\end{cases}$

誤差 $ze_i$, $zl_i$ を求めるにはまず $ze_d$, $zl_i$ を求め，$ze_d$ を整数化して $ze_i$ を求めればよい．乗除算の場合には指数部と仮数部を分離して演算できるので，$(xm_d, xe_d, xl_i)$ と $(ym_d, ye_d, yl_i)$ から計算でき，加減算では (11) 式に $x$, $y$ がないので $(xp_i, xe_d, xl_i)$ と $(yp_i, ye_d, yl_i)$ から計算される．このとき次式で $ze_d\to ze_i$ の変換を行う．

$$ze_i=[ze_d+r_d]+1,\quad [\ \ ]\text{ は Gauss 記号.} \quad (14)$$

配列名

| LXI | ( ,1) | ( ,2) |
|---|---|---|
| (1, ) | 1 | 1 |
| (2, ) | 0 | 0 |
| (3, ) | 3 | 3 |
| (4, ) | 1415926535 | 1415926535 |
| (5, ) | 8979323846 | 8979323846 |
| (6, ) | 2643383281 | 2643383279 |

Fig. 4 Format of I type

ここで $r_d$ は丸めに対する補正項であり，切り捨て，切り上げなら1，四捨五入なら 0.5 となる．(11)，(13)式の右辺は正数の演算であり，減算がないので桁落ちの心配がなく，(14)式で倍精度実数型の演算誤差と (13) 式の近似誤差は充分に保証される．

### 6.4 I型 (interval arithmetic)

実用上，E型でも演算誤差の評価には充分であるが，より完全を期すには interval arithmetic を行う．I型ではS型を基本とし，interval number を二次元整数配列で表わし，図4にその例を示す．

interval arithmetic は次式で定義される．

$$\left.\begin{aligned}&[x_1, x_2]\textcircled{op}[y_1, y_2]=[z_1, z_2]\\&z_2=\max(a_{11}, a_{12}, a_{21}, a_{22})\\&z_1=\min(a_{11}, a_{12}, a_{21}, a_{22})\end{aligned}\right\} \quad (15)$$

ただし，$a_{ij}=x_i\textcircled{op}y_j$，$\textcircled{op}$ は四則演算．

有限桁演算の場合は丸めを考慮して次式となる．

$$\left.\begin{aligned}&z_2=\max([a_{11}]_U,\ [a_{12}]_U,\ [a_{21}]_U,\ [a_{22}]_U)\\&z_1=\min([a_{11}]_L,\ [a_{12}]_L,\ [a_{21}]_L,\ [a_{22}]_L)\end{aligned}\right\} (16)$$

$$[a_{ij}]_U=\begin{cases}a_{ij}\text{ の丸めは切り上げ} & a_{ij}\geqq 0\\ a_{ij}\text{ の丸めは切り捨て} & a_{ij}<0\end{cases}$$

$$[a_{ij}]_L=\begin{cases}a_{ij}\text{ の丸めは切り捨て} & a_{ij}\geqq 0\\ a_{ij}\text{ の丸めは切り上げ} & a_{ij}<0\end{cases}$$

この場合の切り上げは真の切り上げである．真の切り上げを行うには加減乗算では第1，第2 operand で0でない桁の下限に注目し，除算では余りの0判定を行えばよい．(16)式を四則演算のおのおのの場合に適用すると下記のようになる．

加算……$z_1=[a_{11}]_L$，$z_2=[a_{22}]_U$

減算……$z_1=[a_{12}]_L$，$z_2=[a_{21}]_U$

乗算……表1，除算……表2

実際の処理においてI型は数をM型に変換した後，M型の二数に対して四則演算を行って interval arithmetic 用の丸め処理を行う手続き“basic interval”を独立してもっている．

## 7. 演算時間と適用例

以下の計算は大阪大学大型計算機センターのNEA-

Table 1 Multiplication for interval arithmetic (finite length)

| 上段 $Z_2$<br>下段 $Z_1$ | $x_2\leqq 0$ | $x_1\cdot x_2<0$ | $x_1\geqq 0$ |
|---|---|---|---|
| $y_2\leqq 0$ | $[a_{11}]_U$<br>$[a_{22}]_L$ | $[a_{11}]_U$<br>$[a_{21}]_L$ | $[a_{12}]_U$<br>$[a_{21}]_L$ |
| $y_1\cdot y_2<0$ | $[a_{11}]_U$<br>$[a_{12}]_L$ | $\max([a_{11}]_U, [a_{22}]_U)$<br>$\min([a_{12}]_L, [a_{21}]_L)$ | $[a_{22}]_U$<br>$[a_{21}]_L$ |
| $y_1\geqq 0$ | $[a_{21}]_U$<br>$[a_{12}]_L$ | $[a_{22}]_U$<br>$[a_{12}]_L$ | $[a_{22}]_U$<br>$[a_{11}]_L$ |

Table 2 Division for interval arithmetic (finite length)

| 上段 $Z_2$<br>下段 $Z_1$ | $x_2\leqq 0$ | $x_1\cdot x_2<0$ | $x_1\geqq 0$ |
|---|---|---|---|
| $y_2<0$ | $[a_{12}]_U$<br>$[a_{21}]_L$ | $[a_{12}]_U$<br>$[a_{22}]_L$ | $[a_{11}]_U$<br>$[a_{22}]_L$ |
| $y_1\cdot y_2\geqq 0$ | 不能 | 不定 | 不能 |
| $y_1>0$ | $[a_{22}]_U$<br>$[a_{11}]_L$ | $[a_{21}]_U$<br>$[a_{11}]_L$ | $[a_{21}]_U$<br>$[a_{12}]_L$ |

Table 3 Comparison of CPU time

(単位 msec.)

| サブルーチン名 | | 動作 | NEAC 2200 series | | FACOM 230-60 | |
|---|---|---|---|---|---|---|
| | | | model 700 | model 500 | FORTRAN C | FORTRAN D |
| M型 | ADD | M+M | ～1 | 3～6 | 2～3 | 2 |
| | SUB | M−M | ～1 | 3～6 | 2～3 | 2 |
| | MULT 1 | M∗M | 6 | 60～63 | 12～14 | 10～12 |
| | DIVI 1 | M/M | 7～9 | 70～80 | 14～18 | 10～16 |
| | MULT 2 | M∗I | ～1 | 5～6 | 2 | 1～2 |
| | DIVI 2 | M/I | 1 | 5～6 | 2 | 1～2 |
| | CVSHO | M→D | 1 | 4～6 | 1～2 | 1 |
| | CVLNG | D→M | 1 | 8～9 | 1 | 2 |
| S型 | ADDS | M+M | ～1 | 2～6 | 1～3 | 1～2 |
| | SUBS | M−M | ～1 | 2～6 | 1～3 | 1～2 |
| | MULT 1 S | M∗M | 5～6 | 58～60 | 12～14 | 10～12 |
| | DIVI 1 S | M/M | 6～8 | 75～80 | 18～20 | 14～16 |
| | MULT 2 S | M∗I | ～1 | 5～7 | 2 | 1～2 |
| | DIVI 2 S | M/I | ～1 | 5～7 | 2 | 2 |
| | CVSHOS | M→D | ～1 | 5 | 1～2 | 1 |
| | CVLNGS | D→M | ～1 | 8～9 | 1 | 2 |
| E型 | ADDE | M+M | 1～2 | 13～15 | 2～3 | 2 |
| | SUBE | M−M | 1～2 | 13～15 | 2～3 | 2 |
| | MULT 1 E | M∗M | 5～6 | 50～65 | 13～16 | 8～11 |
| | DIVI 1 E | M/M | 6～10 | 85～95 | 20～23 | 15～17 |
| | MULT 2 E | M∗I | 1 | 8～10 | 2 | 1～2 |
| | DIVI 2 E | M/I | 1 | 8～10 | 2～3 | 2 |
| I型 | ADDI | M+M | 3～4 | 25～32 | 7～9 | 5～6 |
| | SUBI | M−M | 3～4 | 25～32 | 7～9 | 5～6 |
| | MULT 1 I | M∗M | 17～18, 35 | 160～173, 335 | 30～34, 62 | 23～25, 45 |
| | DIVI 1 I | M/M | 13～17 | 150～170 | 31～45 | 27～30 |
| | MULT 2 I | M∗I | 3 | 23～27 | 5～9 | 4～5 |
| | DIVI 2 I | M/I | 3 | 23～26 | 5～7 | 4～5 |

動作欄：M…M, S, E, I 型の多重精度実数
I…$|I|<10^n$ の単精度整数
D…倍精度実数

Table 4 Result of example 1 (S type)

```
ROUND OFF PARAMETER ----- R=      0
CPU TIME            ----- 5883 MSEC.

      1          0       27719  99999999999  99999999999  99999999999  99999998761  10035088954
      1          1           3  60360000000            0            0            1  70456448813
      1          1           3  60359999999  99999999999  99999999999  99999999942  98806974690
      1          1           3  60360000000            0            0          815  23276444432
      1          1           7  20719999999  99999999999  99999999999  99999993787  84347454427
      1          1         122  52240000000            0            0        28176  22572276841
      1          1         122  52239999999  99999999999  99999999999  99999919379  52912537787
      1          1        2327  92560000000            0            0       149262  12399090098
      1          1        2327  92559999999  99999999999  99999999999  99998821584  83199606772
      1          1        2327  92560000000            0            0       132885  69544478409
      1          1        2327  92559999999  99999999999  99999999999  99999943928  73299224470
      1          1       53542  28880000000            0            0        10235  23835946660


ROUND OFF PARAMETER ----- R=  50000
CPU TIME            ----- 5862 MSEC.

      1          0       27720            0            0            0         3825  77783153?2
      1          1           3  60359999999  99999999999  99999999999  99999999994  99192129?4
      1          1           3  60360000000            0            0          161  54859817626
      1          1           3  60359999999  99999999999  99999999999  99999997753  58858422558
      1          1           7  20720000000            0            0        16745  63154446?5
      1          1         122  52239999999  99999999999  99999999999  99999923390  26837086?1
      1          1         122  52240000000            0            0       210350  72491240?9
      1          1        2327  92559999999  99999999999  99999999999  99996153?5  84641421??
      1          1        2327  92560000000            0            0       454904  19483624?9
      1          1        2327  92559999999  99999999999  99999999999  99996642?5  17957270?8
      1          1        2327  92560000000            0            0       140530  16278882?7
      1          1       53542  28879999999  99999999999  99999999999  99999974527  79047028?1


ROUND OFF PARAMETER ----- R= 100000
CPU TIME            ----- 5867 MSEC.

      1          0       27720            0            0            0         7559  80063922?1
      1          1           3  60359999999  99999999999  99999999999  99999999990  38553?1865
      1          1           3  60360000000            0            0          303  24318298?9
      1          1           3  60359999999  99999999999  99999999999  99999993858  32578959?5
      1          1           7  20720000000            0            0        30419  42582488?0
      1          1         122  52239999999  99999999999  99999999999  99998866130  74930122?2
      1          1         122  52240000000            0            0       373485  98150047?5
      1          1        2327  92559999999  99999999999  99999999999  99993232?1  42571721?2
      1          1        2327  92560000000            0            0       194187  18224869?4
      1          1        2327  92559999999  99999999999  99999999999  99994174?5  99515238?4
      1          1        2327  92560000000            0            0       442174  82362856?5
      1          1       53542  28879999999  99999999999  99999999999  99999956342  13903188?1
```

Table 5 Result of example 1 (E type)

```
ROUND OFF PARAMETER ----- R=  50000
CPU TIME            ----- 6868 MSEC.

      9   159953670          0       27720     0     0     0     0     0     0     0  3997
      9      201795          1           3 60359 99999 99999 99999 99999 99999 99999 99995
      9     6317956          1           3 60360     0     0     0     0     0     0   169
      9    85722015          1           3 60359 99999 99999 99999 99999 99999 99999 97659
      9   625972897          1           7 20720     0     0     0     0     0     0 17438
      9  2740465180          1         122 52239 99999 99999 99999 99999 99999 99999 22346
      9  7610371251          1         122 52240     0     0     0     0     0     2 18837
      8      137335          1        2327 92559 99999 99999 99999 99999 99999 99996
      8      161329          1        2327 92560     0     0     0     0     0     5
      8      118350          1        2327 92559 99999 99999 99999 99999 99999 99997
      9  4927609440          1       53542 28879 99999 99999 99999 99999 99999 99999 73540
      9   888928464          1
```

C 2200-500, 700, 京都大学大型計算機センターの FACOM 230-60 による結果であるが，特に示さないかぎり NEAC 2200-700 によるものとする．記号 $n$ は仮数部の配列一要素に格納される 10 進の桁数，$l$ は 10 進で表わした演算桁数，$r$ は丸めのパラメータとする．

### 7.1 演算時間の比較

条件…$n=5$, $l=55$, 結果…表 3．

### 7.2 適用例

例 1　ill-condition における試み

$\boldsymbol{AX}=\boldsymbol{B}$ を掃出し法[9]によって解く．

$$\boldsymbol{B}=\begin{pmatrix}549947480\\505269308\\467420948\\434913308\\406674622\\381906956\\360002020\\340487160\\322989141\\307209091\\292904731\\279877507\end{pmatrix},\quad \boldsymbol{X}=_{(\text{正解})}\begin{pmatrix}27720\\360360\\360360\\360360\\720720\\12252240\\12252240\\232792560\\232792560\\232792560\\232792560\\5354228880\end{pmatrix}$$

Table 6 Result of example 1 (I type)

CPU TIME ----- 25863 MSEC.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *** UPPER BOUND *** | 1 | 0 | 27720 | 0 | 0 | 0 | 180857502 | 9504715882 |
| *** LOWER BOUND *** | 1 | 0 | 27719 | 9999999999 | 9999999999 | 9999999999 | 9819136776 | 4033802802 |
| *** UPPER BOUND *** | 1 | 1 | 3 | 6036000000 | 0 | 0 | 228167 | 1900458533 |
| *** LOWER BOUND *** | 1 | 1 | 3 | 6035999999 | 9999999999 | 9999999999 | 9999771840 | 2210681785 |
| *** UPPER BOUND *** | 1 | 1 | 3 | 6036000000 | 0 | 0 | 7143332 | 2137587201 |
| *** LOWER BOUND *** | 1 | 1 | 3 | 6035999999 | 9999999999 | 9999999999 | 9992856430 | 5590155078 |
| *** UPPER BOUND *** | 1 | 1 | 3 | 6036000000 | 0 | 0 | 96922280 | 504248997 |
| *** LOWER BOUND *** | 1 | 1 | 3 | 6035999999 | 9999999999 | 9999999999 | 9903080999 | 5031564270 |
| *** UPPER BOUND *** | 1 | 1 | 7 | 2072000000 | 0 | 0 | 707725865 | 6002258555 |
| *** LOWER BOUND *** | 1 | 1 | 7 | 2071999999 | 9999999999 | 9999999999 | 9292249798 | 2901465415 |
| *** UPPER BOUND *** | 1 | 1 | 122 | 5224000000 | 0 | 0 | 3098470633 | 77673581 |
| *** LOWER BOUND *** | 1 | 1 | 122 | 5223999999 | 9999999999 | 9999999999 | 6901637402 | 7581642096 |
| *** UPPER BOUND *** | 1 | 1 | 122 | 5224000000 | 0 | 0 | 8604057762 | 1644920291 |
| *** LOWER BOUND *** | 1 | 1 | 122 | 5223999999 | 9999999999 | 9999999999 | 1395438545 | 1897663523 |
| *** UPPER BOUND *** | 1 | 1 | 2327 | 9256000000 | 0 | 1 | 5527052412 | 8912712477 |
| *** LOWER BOUND *** | 1 | 1 | 2327 | 9255999999 | 9999999999 | 9999999998 | 4473501544 | 7758029184 |
| *** UPPER BOUND *** | 1 | 1 | 2327 | 9256000000 | 0 | 1 | 8239197867 | 7228131210 |
| *** LOWER BOUND *** | 1 | 1 | 2327 | 9255999999 | 9999999999 | 9999999998 | 1760148312 | 9778718644 |
| *** UPPER BOUND *** | 1 | 1 | 2327 | 9256000000 | 0 | 1 | 3380671056 | 5465290037 |
| *** LOWER BOUND *** | 1 | 1 | 2327 | 9255999999 | 9999999999 | 9999999998 | 6619810627 | 1791420175 |
| *** UPPER BOUND *** | 1 | 1 | 2327 | 9256000000 | 0 | 0 | 5571098905 | 6245472406 |
| *** LOWER BOUND *** | 1 | 1 | 2327 | 9255999999 | 9999999999 | 9999999999 | 4428699768 | 2275834535 |
| *** UPPER BOUND *** | 1 | 1 | 53542 | 2888000000 | 0 | 0 | 1005056227 | 2759516321 |
| *** LOWER BOUND *** | 1 | 1 | 53542 | 2887999999 | 9999999999 | 9999999999 | 8994980217 | 3570965814 |

***A***…12 次 Hilbert 行列

条件…$n=5$, $l=55$, 結果…表 4～6

例 2 解を上下限で決定する試み

(A) $\pi$ について

$$\pi=48\,\mathrm{Tan}^{-1}\frac{1}{18}+32\,\mathrm{Tan}^{-1}\frac{1}{57}-20\,\mathrm{Tan}^{-1}\frac{1}{239}$$ [10]

$$\mathrm{Tan}^{-1}\frac{1}{x}=\sum_{m=1}^{\infty}a_m,\qquad a_m=\frac{(-1)^{m-1}}{(2m-1)\,x^{2m-1}}$$

$\pi$ を区間で決定するには逆正接を区間で表わす必要があるので次式で計算を行った.

$$\sum_{m=1}^{\text{even}}a_m<\mathrm{Tan}^{-1}\frac{1}{x}<\sum_{m=1}^{\text{odd}}a_m.$$

結果…表 7.

(B) Newton 法について

Newton 法で根の上下限を決定するには interval version[3] があるが, ここでは加速を考えて通常の Newton 法をもちいる. その例として(5)式により $\sqrt{a}$ を求める場合を考える. (6)式の右辺が常に正なので丸めに注意さえすれば (5) 式を求めた根は真値より大きくなる. すなわち, (5) 式で求めた平方根を $\sqrt{a}'$ で表わすと次のようになる.

$$\frac{1}{\sqrt{\frac{1}{a}}'}\leqq\sqrt{a}\leqq\sqrt{a}'.\qquad(17)$$

結果…表 7.

(単位 msec.)

Table 7 Result of example 2

| example | | 2-A ($n$=7) | | 2-B ($n$=5) | |
|---|---|---|---|---|---|
| $l$ | | 112 | 1057 | 55 | 205 |
| 精度 | 可変 | 665 | 31986 | 106<br>254 | 1047<br>2312 |
| | 固定 | 765 | 51153 | 108<br>269 | 1406<br>3327 |

ex. 2-B欄 {上段……(17)式<br>{下段……interval version

## 8. むすび

解の存在が保証されていて, かつ近似解法でない場合は無限桁演算を行えば ill-condition の場合でも解が求まるものと思われる. したがって, これらの問題に対しては実用上, 演算桁数を長くとることによって解決できることが多い. このような例として例 1 を示した. Hilbert 行列は ill-condition とされており, 12 次元程度以上では単精度, 倍精度では解けないが, これは演算桁数が短いからであり, 本質的なものでないことは表 4～6 よりわかる. 表 4 によれば丸めの変化による解の変動は少ない. これは S 型のもつ機能である丸めの制御ができることの一例である. 表 4 より表 5, 6 のほうが解の広がりが大きいが, 誤差評価の手段が異なるので当然の結果である. 直接的解法に対して E, I 型が有効であるのは表 5, 6 からわかり,

特にI型による解はその区間内に必ず真値をふくみ，演算誤差の評価を行うことができる．

一方，反復法の中で Newton 法のように前回の演算誤差が波及しない場合には，M，S，E，I型のいずれの場合にも **4.** で述べた演算時間の短縮ができる．このような場合にはI型では反復を有限回で打切るための打切り誤差をもふくめて，解の存在範囲を決定しうる場合がある．E型では (11)，(13) 式を実際に計算する時に実数型計算を行っているのと，(13)式が近似式であることにより $x \pm \delta x$ の範囲から値がはずれる可能性がある．この可能性は非常に小さく，実用上の問題点はほとんどないものと思われるが，このためすでに上に述べたI型で反復法を行う例のような打切り誤差をもふくめた解の存在範囲を決定することは，論理的に不可能となる．しかし，各ステップにおける演算誤差が求められるので，反復の停止則などの判定に使用できる．また，アルゴリズムを適当にえらべば，直接に interval arithmetic を適用しなくても解を上下限ではさむことができる．そして，例2のように interval arithmetic を適用しても意味のない問題や，あるいは，演算時間の短縮に効果がある．この場合の演算は，**7.4** で述べた $[\ ]_U$，$[\ ]_L$ の丸めを行う必要があるので“basic interval”で行っている．

以上に述べたごとく，誤差評価機能をもった多重精度演算の存在意義は大きく，多くの目的においてその成果を期待しうるものである．

終りに，日頃ご指導をいただく大阪大学工学部久保忠雄教授に深謝いたします．

## 参考文献

1) M. Goldstein & S. Hoffberg: The Estimation of Significance, In J. R. Rice (Ed.), Mathematical Software, pp. 93～104, Academic Press, New York (1971).
2) C. V. Ramamoorthy & M. Tsuchiya: Micrprogramed Significance Arithmetic-A Perspective and Feasibility Study, Proc. of SJCC, Vol. 40, pp. 659～673 (1972).
3) R. E. Moore: Interval Analysis, p. 145, Prentice-Hall, Englewood Cliffs, New Jersey (1966).
4) D. I. Good & R.L. London: Computer Interval Arithmetic-Definition and Proof of Correct Implementation, J. ACM, Vol. 17, No. 4, pp. 603～612 (1970).
5) たとえば，I.D. Hill: Algorithm 34-Procedure for the Basic Arithmetical Operations in Multiple-Length Working, Comput. J., Vol. 11, No. 2, pp. 232～235 (1968).
6) P. Rabinowitz: Multiple-Precision Division, Comm. ACM, Vol. 4, No. 2, p. 98(1961).
7) E. V. Krishnamurthy: On a Divide-and-Correct Method for Variable Precision Division, Comm. ACM, Vol. 8, No. 3, pp. 179～181 (1965).
8) C. J. Mifsud: A Multiple-Precision Division Algorithm, Comm. ACM, Vol. 13, No. 11, pp. 666～668 (1970).
9) 雨宮綾夫，田口武夫編：数値解析と FORTRAN, p. 469, 丸善，東京 (1969).
10) D. Shanks & J.W. Wrench, Jr.: Calculation of $\pi$ to 100,000 Decimals, Math. Comp., Vol. 16, No. 77, pp. 76～99 (1962).
11) 大中幸三郎，安井　裕：誤差評価の可能な多重精度演算について，京都大学数理解析研究所講究録 172, pp. 119～137 (1973).

(昭和48年4月23日受付)
(昭和48年9月10日再受付)