解　説

7. パソコンの精度保証付き数値計算ソフトウェア†

野　寺　　隆††

1. はじめに

数値計算の歴史は，古く太古の昔にさかのぼることができる．しかし，電子計算機が現れるにいたり，その形態もさまざまに変化したように思われる．特に，浮動小数点演算が採用されるに従って，数値計算の精度の保証が重要な要因となってきた．また，数値の表現方式に関しても，近年，IEEE 方式[7]，松井，伊理方式[10]，浜田方式[4),5)]などさまざまなものが提案されている．

Moore[11] によって提案された区間演算法は，精度の保証を考えた演算方式の一つである．区間演算法が開発された当時，我が国にも紹介されたが，アイデアがあまりにも素朴で，その効果が思ったほど上がらなかったこともあり，この演算方法を否定する人々が多かった．しかし，近年，西欧を中心とした区間数学の研究が広がり，そのアルゴリズムも改善され再び脚光をあびてきた．また，従来，区間演算ソフトウェアは特殊な計算機でしか利用できなかったのだが，近年のパーソナルコンピュータ（パソコン）の急速な普及により，簡単に身近で利用できるものも現れてきた．

PASCAL-SC (PASCAL for Scientific Computation) は，区間演算をパソコンで利用できるソフトウェアの代表的なものであり，この分野の研究者にとってはなくてはならないものである．

本稿では，PASCAL-SC を中心として，区間演算ソフトウェアの機能と性能の特徴を解説し，現在のパソコン用区間演算ソフトウェアの研究状況とその動向について解説するものである．

最後に，PASCAL-SC の区間演算パッケージを利用して簡単な代数方程式の解を計算する数値例をあげることにする．

† Self-Validating Computational Software for Personal Computers by Takashi NODERA (Department of Mathematics, Faculty of Science and Technology, Keio University).

†† 慶應義塾大学理工学部

2. PASCAL-SC

パソコンでも利用できる PASCAL-SC を紹介する前に，なぜこのようなソフトウェアが必要なのか，簡単な例題で考えてみることにしよう．

2.1 なぜ区間演算か

浮動小数点演算を採用して有限桁の四則演算（+，−，×，／）をすると，必ずといってよいくらい丸め誤差，情報落ち，あるいは桁落ちなどの問題を引き起こすことになる．しかし，単純な浮動小数点演算では，得られた結果がどの程度の精度まで正しいのか判断するのはなかなか厄介である．たとえば，次のような簡単な例を考えてみることにしよう．

［例］ 関数 $f(x,y)=9x^4-y^4+2y^2$ が与えられたとき，$x=10864$, $y=18817$ の場合に $f(x,y)$ の値を計算せよ．

通常，この数式を計算するには，数値計算の定石として，次の三つの計算方法が考えられる．

（1） f:=9*x**x*x*x*x-y*y*y*y
　　+2*y*y;

（2） f:=9*(x**4)-y**4+2*(y**2);

（3） f:=(3*(x**2)-y**2)*(3*(x**2)
　　+y**2)+2*(y**2);

すなわち，(1)は単純に数式のとおりに記述したもの，(2)はベキ乗の計算を先に行い，それに定数を掛けるもの，(3)は数式を因数分解して計算するものである．このおのおのの数式を13桁の十進演算を行うと，

（1） $f=1.58978\times10^5$,

（2） $f=8.41022\times10^5$,

（3） $f=1.000000000000$

という解を得ることになる．当然，おのおのの演算は，最も近い浮動小数点数に丸められている．

もうすでにお分かりのことと思うのだが，正しい結果は1である．このように単純な浮動小数点演算を用いると，計算方法によって解にこんなに差が現れてしまうことになる．考えてみれば，恐ろしいことであ

る．しかし，数値計算を多少なりとも勉強していると，f の値を(3)式で計算するのが最もよいことが分かる．逆に，$x=18817$, $y=10864$ として，f の値を上の三つの方法で計算すると，今度は三つの数式とも同じ値

$$f=1.11442030725\times10^{18}$$

となるのである．これはほぼ正確な値が計算されている．この例のように，与えられるデータによっては関数の値を計算する数式が違っていても，どれも正確な値が計算されることもある．

当然，このような問題では，従来からある誤差解析を用いることによって丸め誤差を見積もることができる．しかし，広範囲にわたる数式を取り扱う場合には，従来型の丸め誤差解析によって誤差を推定することはあまり便利でなく，もしそれを用いたとしても多大なる注意を払わなくてはならない．そんな場合に，区間演算を用いると，数値の誤差限界の計算をわりに簡単に行うことができる．

区間演算の基本は，閉じた有限の実数の区間 $[a,b]$ を用いて演算を行うのである．この区間は，下限が a で上限が b の実数の集合を示している．当然，上限と下限は等号を含むことになる．

区間演算は，科学技術計算だけでなく，統計計算においても広範に応用することができるのだが，今日ではそれほど広く利用されていない．その理由の一つは，計算機の演算方式と同様に，汎用性の高いプログラミング言語さえ区間演算をサポートしていないからである．

計算機の演算方式の一般理論の中で，区間演算は実数のベキ集合における演算の特殊な場合として導入することができる．すなわち，もし X と Y が R の部分集合で，$\circ\in\{+,-,*,\times\}$ ならば，定義によって

$$Z=X\circ Y=\{x\circ y\mid x\in X, y\in Y\}$$

となる．ただし，割算に関しては，$0\in Y$ は当然除外されている．もし，$X=[a,b]$, $Y=[c,d]$ が区間であれば，$Z=[r,s]$ も定義され，これも区間となる．さらに，Z の下限 r と上限 s は，X と Y の下限と上限の値 a,b,c,d から計算することができるのである．もちろん，区間 $X=[a,b]$ は，$a\leq b$ でなければいけない．

たとえば，前記の関数 f の値を区間 X と Y における区間演算を用いて計算するものとすれば，得られた区間は，$\forall x\in X$ と $\forall y\in Y$ に対して関数 f のすべての値を含むことになる．この区間演算の性質は，最終値まで詳細な解析をしなくても，関数の値域の限界を計算できることである．浮動小数点を用いた区間演算では，常に正確な結果を含む最も小さな浮動小数点の区間へ，区間演算の結果を丸めることになる．そこで，数式の値を評価する丸め誤差の限界は，初期値として点区間 $X=[x,x]$ と $Y=[y,y]$ をとることによって都合よく計算できるのである．

ここで，前述の x と y を $X=[10864, 10864]$ と $Y=[18817, 18817]$ の区間に置き換えて f の値を計算すると，

(1′) $F=[-1.841022\times10^{6}, 1.158978\times10^{6}]$,

(2′) $F=[-8.410220\times10^{5}, 1.158978\times10^{6}]$,

(3′) $F=[1.000000000000, 1.000000000000]$

となる．この結果を見ると，(1′)と(2′)の F の値は，上限と下限の間の幅が大きいので，得られた解に信用がおけないことが分かる．すなわち，これは丸め誤差が大きいことを示している．また，これに対して，(3′)の結果は，精度が高いことを示している．事実，真の値は，1なのだから．このように，区間演算を用いると，丸め誤差の影響を見積もることができ，解の精度保証にも役立つことになる．このように，区間演算も簡単にできるシステムとして，PASCAL-SC をあげることができる．

2.2 PASCAL-SC とは

PASCAL-SC が最初に発表されたのは，1980 年の4月にドイツのハノーバー (Hannover) で開催されたハノーバー・フェアー (Hannover Fair) であった．このシステムを制作したのは，カールスルーエ (Karlsruhe) 大学の応用数学研究所であった．当時のシステムは，今販売されているものと比較してもそれほどの見劣りのしないものであった．それ以後，システムにいろいろな改良や機能が追加され，Z 80, 8088, 68000 のプロセッサで利用できるようになり，PASCAL-SC はおもに IBM-PC/AT のパソコン用の OS である MS-DOS 上に移植されている．現在，最も使われている PASCAL-SC は，この MS-DOS 上のものであろう．また，マッキントッシュ (Macintosh) で動くものもある．

PASCAL-SC の開発理念は，次の三つに集約される．

1. マイクロコンピュータの驚異的な発展により，高級言語のプログラミングが可能になった．

2. マイクロコンピュータでも数値解を計算できるアルゴリズムの開発が進んできた．

3. PASCAL は，科学技術計算を行う言語として教育しやすく，豊富な機能を提供してくれる．

当然，PASCAL-SC は，PASCAL という名前が付いているからには，標準的な PASCAL としての機能をもっている．そのほかに，次のような機能をあげることができる．

- シンタックスをチェックしたり，コンパイルエラーをすばやく報告するためのスクリーンエディタがある．また，自動インデンテイションの機能をもっている．
- 一般的な演算子の概念がある．
- 任意結果型の関数がある．
- 抽象データ型に対する演算子の概念がある．
- 十進 12 桁（最大精度）の浮動小数点演算ができる．
- 直接丸めや区間演算の意味で誤差の制御ができる．
- 最大精度の行列計算ができる．
- 線形計算や固有値，固有ベクトルの計算で得られた答の精度保証ができる．
- INCLUDE 機能がある．
- プログラムを分割コンパイルすることができる．
- 実行時にソース行を参照するエラーメッセージを出力できる．
- 手軽にプログラミング教育やデータ処理に利用できる．
- IBM-PC のカラーグラフィックスに対するグラフィックインタフェースが準備されている．
- 簡単なプロット機能がある．
- MS-DOS の機能にアクセスできる．

これらの機能に加え，PASCAL-SC に与えられている浮動小数点演算は，実数だけでなく複素数や実区間，複素区間，さらに行列やベクトルの計算にも用いることができる．すなわち，PASCAL-SC は，通常の PASCAL に比べて，より科学技術計算向きに作られたシステムであり，数値アルゴリズムを非常に簡単に記述できるようになっている．さらに，数値解析に共通する問題解決に必要ないくつかの応用パッケージが準備されている．また，問題の条件が悪い場合にも，かなり精度の高い解を自動的に得ることができたり，解の精度を自動的に見積もる機能も備えている．

PASCAL-SC の浮動小数点演算は，Kurisch と Miranker[9] によって提案された計算機の演算理論に基づいて作られている．また，その演算方式は結果の精度，制御性や信頼性を保証するものである．

さらに，マイクロコンピュータの上で作動するように，PASCAL-SC のコンパイラをできるだけ小さく作りあげている．さらに，コンパイラはユーザプログラムとリンクできるように，外部ライブラリを広範に利用するように作られている．

PASCAL-SC は，もう一つ次のような重要な機能をもっている．それは，ユーザが自分で演算子を定義できることである．たとえば，行列を A とし，$\boldsymbol{b}, \boldsymbol{y}, \boldsymbol{x}$ をベクトルとし，

$\boldsymbol{y} := A * \boldsymbol{x} + 3 * \boldsymbol{b}$;

と記述すれば，PASCAL-SC は行列 A とベクトル $\boldsymbol{x}$ の積を計算し，それにスカラ 3 とベクトル $\boldsymbol{b}$ の積を計算し，その値をおたがいに加え，その結果をベクトル $\boldsymbol{y}$ に代入することになる．すなわち，この場合の＊記号は，行列とベクトルの積を表し，＋記号はベクトルとベクトルの和を計算するものと解釈されるのである．

3. 区間演算機能について

PASCAL-SC の機能として区間演算をあげることができる．そこで前章でも簡単に述べたが，区間演算についてもう少し詳しく述べることにしよう．区間演算を一言で述べれば，すべてのデータをある幅をもった数値として考え，計算のステップでその結果の幅を計算することである．

最初に区間演算の簡単な規則について述べることにする．いま，二つの区間を $X=[a,b]$ と $Y=[c,d]$ としよう．まず最初に，

$$+I=+[a,b]=[a,b]$$
$$-I=-[a,b]=[-b,-a]$$

を定義すると，2 項演算は

$$X \circ Y=[a,b] \circ [c,d]=\{x \circ y \mid x \in X, y \in Y\}$$

と定義すればよいことになる．ただし，$\circ \in \{+,-,\times,/\}$．

割り算は，区間 $Y=[c,d]$ が 0 を含まないときに定義することができる．当然，X と Y が区間であれば，$X \circ Y$ も区間となり，この区間の両端の値は X と Y の区間の上限と下限から次のように計算することができる．

$$\begin{aligned} X \circ Y &= [a,b] \circ [c,d] \\ &= [\min\{a \circ c, a \circ d, b \circ c, b \circ d\}, \\ &\quad \max\{a \circ c, a \circ d, b \circ c, b \circ d\}] \end{aligned}$$

この数は，一見すると複雑そうに見えるが，和と差に関していえば非常に単純である．

$$X+Y=[a,b]+[c,d]=[a+c,b+d]$$

$$X-Y=[a,b]-[c,d]=[a-d,b-c]$$

ただし，注意しなければならないのは，$[a,b]-[a,b]=[a-b,b-a]$ は，一般に $[0,0]$ でないことである．このように区間演算の性質は，一般の実数計算の性質と異なる場合が多い．

次に掛け算の場合であるが，これは a,b,c,d の符号を解析することによって単純化することができる．たとえば，$a\geq 0, c\geq 0$ とすれば，

$$X*Y=[a,b]*[c,d]=[a*c,b*d]$$

となる．掛け算の可能性は，9つの場合があることが分かると思うが，これは単純に二つの数を掛け合わせるよりも多くの計算を必要とする．

割り算に関しては，掛け算と同じように考えればよい．ただし，$c*d\leq 0$ となる三つの場合を除くので，6通りの計算をする可能性がある．

ここまでの区間演算の定義は，正確な実数演算を仮定して述べてきた．しかし，実際に計算機を利用してこのような区間演算を行うには，区間演算そのものが有限桁の浮動小数点演算で行われるので，計算途中の丸め誤差はその最後の1ビットに関するものである．そこで，下限のほうの値は常に切り捨て，上限のほうの値は常に切り上げる処理をして計算すればよい．すなわち，二項演算を次のように定義すればよいことになる．

$$X\circ Y=[a,b]\circ[c,d]$$
$$=[\min\{a\circ<c, a\circ<d, b\circ<c, b\circ<d\},$$
$$\max\{a\circ>c, a\circ>d, b\circ>c, b\circ>d\}]$$

通常，区間演算を行うには，このような演算を定義し実行する必要がある．しかし，これをユーザが各自プログラムの中で定義し処理するには，かなりのテクニックを必要とするし，それを要求するのもかなりの無理がある．

そこで登場したのが PASCAL-SC である．PASCAL-SC は，区間演算用のパッケージをすでに備えており，区間演算は PASCAL の型定義文で簡単に記述することができる．たとえば，実浮動小数点の区間演算の定義は，

```
type interval=record inf, sup: real end;
```

とすればよい．また，区間演算の和 (addition) の演算子は，次のように宣言することができる．

```
type interval=record inf, sup: real end;
  ...
operator+(a, b :interval)res:interval;
  var c :interval;
  begin c:inf:=a.inf+<b.inf;
        c.sup:=a.sup+<b.sup;
        res:=c
  end {operator+};
```

このような区間演算の和を行うには，通常の和の演算子と同様に

```
z:=x+y;
```

のように記述すればよい．ただし，x,y,z は区間演算型であることを宣言する必要がある．当然，PASCAL-SC には，このような演算子は区間演算モジュールのサブルーチンとして提供されている．

PASCAL-SC のコンパイラをできるだけ小さく有効に作動するために，数値データ型は PASCAL-SC の中に埋め込まれていない．その代わりに，ある演算パッケージが複素数，実区間数や複素区間数に対する最適な演算を，さらにベクトルや行列計算に対する最適な演算も提供するようになっている．

次の表-1 は，すでに定義されている型の名前と，それに対応するデータ構造，およびライブラリの名前を記述したものである．ただし，すでに定義されている演算子や入出力の手続き，変換関数や標準関数などが結合されている．表-1 の中で，上から3行目までのデータタイプは，すでにコンパイルされたものが準備されているので，必要なときにはこれらをリンクして用いればよい．また，これらのソースファイルも提供されているので，'$include' 文を用いてプログラムの中に指定してコンパイルすることもできる．たとえば，

表-1 区間演算に関するデータ型と構造

型	データ構造	ライブラリ
interval	record inf, sup: real end	interval.lib
complex	record re, im p: real end	complex.lib
cinterval	record ire, iim: interval end	cinterval.lib
rvector	array [dimtype] of real	rmatrix.pak
ivector	array [dimtype] of interval	rmatrix.pak
cvector	array [dimtype] of complex	rmatrix.pak
civector	array [dimtype] of cinterval	rmatrix.pak
rmatrix	array [dimtype] of rvector	rmatrix.pak
imatrix	array [dimtype] of ivector	imatrix.pak
cmatrix	array [dimtype] of cvector	cmatrix.pak
cimatrix	array [dimtype] of civector	cimatrix.pak

```
program complex-numbers(input, output);
$include complex.pak
var   x, y : complex;
begin cread(input, x); cread(input, y);
      x := x + y;
      cwrite(output, x); writeln
end.
```

と記述すればよい.

通常，レベル0の PASCAL では，ダイナミックな配列を利用することができないので，この表の中で4行目以降のデータタイプは，コンパイルされたルーチンが提供されていない．そこでこれらのデータタイプを使用するには，必ずベクトルや行列の次元を定義し，対応するパッケージのソースを $include 文を用いて本文のテキストの中に入力する必要がある．たとえば，

```
...
const dim=...
type dimtype=1..dim;
$include imatrix.pak
...
```

と記述すればよい.

3.1 デモプログラム

PASCAL-SC には，いくつかのデモプログラムが準備されており，標準的な数値計算であればこれを使うことができる．当然，区間演算も利用できるようになっている.

- 最適なスカラ積の計算
- 連立一次方程式の解の計算
- 逆行列の計算
- 多項式の計算
- 固有値，固有ベクトルの計算
- 有理関数の評価
- 線形計画法
- プロットのデモ

このデモプログラムはメニュー方式になっており，必要なものを順次指定して利用することができる．逆行列の計算のデモの中には，ヒルベルト行列の逆行列を計算できるものがあり，行列の次元を指定すれば，区間演算を利用して解を求め，それを評価してくれる.

なお，プロットのデモプログラムは，シリアルプリンタ用のものなので，クオリティは今一つである.

4. 数 値 例

PASCAL-SC の区間演算機能を用いて，

$f(x)=x^3-r$

の実数の3乗根をニュートン法を利用して計算する例をあげることにしよう．アルゴリズムは，次のようになる.

$X_{n+1}=(x_n-(x_n^3-r)/(3\cdot X_n^2))**X_n$

ただし，X_n の中点を x_n で表し，**は区間の共通部分をとる演算子である.

区間演算を用いて計算を行うので，x_n や r は，$r=[r,r]$ のように点区間で示される．実際に，PASCAL-SC のプログラムは，次のように記述することができる.

```
program cuberoot(input, output);
  {Calculates an interval inclusion for the
   cube root of a given real number}
$include interval.pak
    {Makes interval arithmetic available}
  var r : real;{The radicand}
      n : interval;{Interval radicand}
      i, j : interval;{Inclusion}
      m : interval;{Midpoint}
      c : char;{Control variable}
  function mid(i : interval) : interval;
    {Calculates the interval-valued
     midpoint of an interval}
     begin
       mid := (input(i.inf)+input(i.sup))/2
     end;
  begin
    c := 'Y';{Set control variable to Yes}
    while c='Y' do
    begin{Interval iteration}
      writeln('Enter Radicand :');
      read(r);
      n := input(r);
      if r=0 then
      writeln('Cube Root=0')
      else
      begin{r<>0}
        writeln('Enter Interval Inclusion of
          Cube Root :');
      iread(input, j);{Inclusion}
```

```
      repeat
      i:=j;
        writeln('i=');iwrite(output,i);
      writeln;
        m:=mid(i);{Midpoint}
        j:=m-(m*isqr(m)-n)/(3*isqr(i));
          {Inclusion}
      if(i><j) or (j=i);
        writeln('No Cube Root In Given
        Interval!')
      elsej:=i**j
      until(i><j)or (j=i);
    end;{r<>0}
      writeln('Another Cube Root(Y/N)?');
      readln;read(c){Control variable}
    end {Interval iteration}
  end.
```

このプログラムを PASCAL-SC でコンパイルとして実行してみよう．たとえば，

1. $f(x)=x^3+3$
2. $f(x)=x^3+5$

の実根を区間演算で計算すると次のようになる．

```
Enter Radicaid:
*-3
Enter Interval Inclusion of Cube Root:
*[-3, -1]
I=[          -3.0E+00,          -1.0E+00]
I=[          -1.9E+00,          -1.0E+00]
I=[          -1.48E+00,         -1.42E+00]
I=[          -1.4428E+00,       -1.4419E+00]
I=[       -1.44224962E+00,      -1.44224953E+00]
I=[    -1.44224957031E+00,     -1.44224957030E+00]
Another Cube Root (Y/N)?
*Y
Enter Radicaid:
*-5
Enter Interval Inclusion of Cube Root:
*[-3, -1]
I=[          -3.0E+00,          -1.0E+00]
I=[          -1.9E+00,          -1.0E+00]
I=[          -1.9E+00,          -1.6E+00]
I=[          -1.72E+00,         -1.70E+00]
I=[       -1.709982E+00,        -1.709971E+00]
I=[    -1.70997594668E+00,     -1.70997594667E+00]
Another Cube Root (Y/N)?
*Y
Interval Radicand
*-5
Enter Interval Inclusion of Cube Root:
*[-100, -1]
I=[          -1.0E+02,          -1.0E+00]
I=[          -4.7E+01,          -1.0E+00]
I=[          -2.2E+01,          -1.0E+00]
I=[          -1.1E+01,          -1.0E+00]
I=[          -5.1E+00,          -1.0E+00]
I=[          -2.8E+00,          -1.0E+00]
I=[          -1.9E+00,          -1.3E+00]
I=[          -1.78E+00,         -1.68E+00]
I=[          -1.712E+00,        -1.709E+00]
I=[       -1.7099762E+00,       -1.7099757E+00]
I=[    -1.70997594669E+00,     -1.70997594666E+00]
ANOTHER CUBE ROOT (Y/N)?
*N
```

これは MITSUBISHI のパソコン MAXY を利用して計算したものである．このように区間演算によって簡単に解を求めることができる．ただし，空の共通部分に遭遇したときには，初期値として用いた区間に解を含んでいないことになるので，プログラムがメッセージを出力するようにしてある．また，前述のプログラムの中で j:=〈数式〉の行がニュートン法の反復公式を記述しているところであるので，この箇所を変更すれば，このほかの関数の解を求めることができる．

5. おわりに

数値計算の精度保証をする区間演算をパソコンで簡単に利用できる PASCAL-SC について概観してきた．このシステムは，コンパクトに作られているし，ソフトウェアの値段もお手頃なので（1万円ぐらい），これからいっそうの活用が望まれる．しかし，数値計算の精度を保証するようなお手軽なソフトウェアがもっと現れてよいように思う．ただし，区間演算にはこれから克服しなければならない問題も数多くあるので，ソフトウェアの開発と同時にアルゴリズムの改良も重要なポイントになるであろう．

最後に，区間演算やその応用に関する多くの例が，文献 1), 12), 8) の本に掲載されているので参考にしてほしい．

参考文献

1) Alefeld, G. and Herzberger, J.: Introduction to Interval Computations, Academic Press, Inc. (1983).
2) Allendörfer, U. and Kirchner, R.: PASCAL-SC User Guide, B. G. Teubner Stuttgart (1987).
3) Bohlender, G., Ullrich, C., von Gudenberg, J. W. and Rall, L. B.: PASCAL-SC, Academic Press, Inc. (1987).
4) 浜田穂積：二重指数分割に基づくデータ長独立実数値表現法，情報処理学会論文誌，Vol. 22, No. 6, pp. 521-526 (1981).
5) 浜田穂積：二重指数分割に基づくデータ長独立

実数値表現法Ⅱ，情報処理学会論文誌，Vol. 24, No. 2, pp. 149-156 (1983).
6) Jensen, K. and Wirth, N.: Pascal User Manual and Report, ISO Pascal Standard. 3rd ed. Springer, New York, 1985 (PASCAL (原書第3版)，原田賢一訳，培風館).
7) Kahan, W. and Palmer, J.: On a Proposed Floating-Point Standard, ACM SIGNUM Newsl., Special Issue, pp. 13-21 (1979).
8) Kaucher, E., Kulisch, U. and Ullrich, Ch. (Eds.): Computer Arithmetic, B. G. Teubner Stuttgart (1987).
9) Kurisch, U. and Miranker, W. L.: Computer Arithmetic in Theory and Practice, Academic Press, Inc. (1981).
10) 松井正一，伊理正夫：あふれのない浮動小数点表示方式，情報処理学会論文誌，Vol. 21, No. 4, pp. 306-313 (1980).
11) Moore, R. E.: Interval Analysis, Prentice-Hall, Englewood Cliffs, N. J. (1966).
12) Moore, R. E.: Methods and Applications of Interval Analysis, SIAM, Philadelphia (1979).

(平成2年4月11日受付)