解　説

6. 大型計算機向き汎用精度保証付き数値計算ソフトウェア†

奥田　晃††　棚町芳弘†††

1. はじめに

数値計算の結果の精度をどのように評価すればよいかという問題は，古くて新しい．コンピュータによる数値計算が進展していくなかで，常に問題として提起され研究されてきた．J. H. Wilkinson の backward error analysis などは，その典型的な成果の一つである[1]．こういった成果の上に数値的に安定した良いアルゴリズムが開発されてきた．しかしその一方で，実際に得られた結果の精度を具体的にどのように定量的に評価すればよいかという問題は，正面から取り組めなかったのも事実である．数値計算の現場では，倍精度計算なら10進で16桁はあるから結果はまず大丈夫だろうとか，心配なら4倍精度で計算すればよいだろうといった「楽観」があり，逆にその裏がえしに，D. E. Knuth が指摘するように[2]，出てきた結果は1桁も信頼できないかもしれないと考える「悲観」もある．

計算結果の精度保証をコンピュータ自身に行わせるという発想とそのための理論的基礎づけは，相当以前からなされてきていたが[3),4)]，それが実際的な意味をもつようになったのはごく最近のことである．コンピュータの計算能力の向上，品質保証の必要性の高まり，理論・道具だての蓄積があいまって，計算結果の精度の評価自身が現実の計算課題としてとりあげられるようになった．

この解説は，精度保証付き数値計算のために開発された IBM のプログラム・プロダクトである高精度演算サブルーチン・ライブラリ ACRITH (High-Accuracy Arithmetic Subroutine Library)[5),6)] について，その機能と構成，応用例を述べるものである．ACRITH は，数値計算での基本的なもの，たとえば連立1次方程式の求解などの問題解決サブルーチンを最上位として，これを支える基本演算ルーチンを次のレベルに，そして最下位に完全精度で計算を行うアキュムレータ関連のルーチンをもつ高精度数学サブルーチン・ライブラリを基本として構成されている．

ACRITH は，西ドイツの Kahlsruhe 大学の応用数学研究所と IBM のワトソン研究所や西ドイツの研究開発部門との共同研究を土台として開発された．1983 年に第1版・リリース1が，1986 年にリリース3が発表されている．現在では，この ACRITH を土台とした上位のインタフェースとしての FORTRAN-SC (FORTRAN for Scientific Computation)[7),8)] が，同じく Kahlsruhe 大学と IBM の共同研究により研究用プロトタイプ・プロダクトとして開発され，使用研究されている．

こういったソフトウェア開発の前提となる理論的基礎づけや方法論，実用化の全般については，文献 9), 10) や，この特集号の各解説に詳しい．ここでは ACRITH の紹介に必要なことがらを概観しておく．

一つの連立方程式 $Ax=b$ の計算解 $\tilde{x}$ が得られたとしよう．たとえばガウス消去法を用い，部分ピボッティングを行い，倍精度で計算したとする．ここで次の問題が基本的である．

「得られた解 $\tilde{x}$ はどこまで正しいか，何桁まで正しいかを具体的に示せ．」

このきわめて自然な問いに対して，直接に答える方法・ツールがごく最近までわれわれの手もとになかったというのが現実であった．この問いに答えるためには次のことがらが必要になる．

(1)　計算解は「点」数値ベクトル $\tilde{x}$ としてでなく，「区間」数値ベクトル「$\tilde{x}_1, \tilde{x}_u$」として示し，この区間内に真の解 x^* が存在することが保証されているようにする．この区間を保証区間とよぶ．

† General Purpose Self-Validating Numerical Computation Software for Large Computer Systems by Akira OKUDA (Tokyo NIC Center, IBM Japan Ltd.) and Yoshihiro TANAMACHI (NIC Marketing Center, IBM Japan Ltd.).

†† 日本アイ・ビー・エム(株) 東京 NIC センター
††† 日本アイ・ビー・エム(株) NIC マーケティング・センター

(2) この保証区間は十分狭いもの，すなわち十分に精密でなくてはならない．ただし，問題が悪条件の場合には，保証区間が求まらなかったり，幅 $(\bar{x}_i - \underline{x}_i)$ が十分狭くならないことが起こる．逆にこのことから，問題の悪条件性を指摘するようにする．

また，これらが可能ということになると，入力誤差の問題にも答えることができるはずである．すなわち，不確定データを区間データとして入力したり，コンピュータへのデータの入力の変換誤差を考慮できるなど，データを区間として与える場合を取り扱うことができる．

先の二つの要求に対しては，また次のことが必要になる．

a．保証区間を求めるアルゴリズム

b．区間演算

c．高精度演算を実現するメカニズム

この三つの要請は，連立1次方程式の場合に限らず，他の問題解決に対しても要求される共通のものである．

ACRITH は，これまでの研究成果を土台として，これらのa，b，cを一つの実用的ツールにまとめあげたものである．この解説の 2. では，これらのa，b，cがどのように具現化されているかを，要となるアルゴリズムとライブラリとしての機能と構成を示して説明する．3. では，実際の数値計算例によって ACRITH の機能をみることにし，あわせて FORTRAN-SC による例も紹介する．

2. ACRITH の機能と構成

2.1 ACRITH の構成要素

ACRITH[5),6)] は，図-1 に示すように三つの要素，高精度数学サブルーチンライブラリ (High-Accuracy Mathematical Subroutine Library)，高精度演算命令 (High-Accuracy Arithmetic)，オンライン学習用モジュール (Online Training Component) からなる．

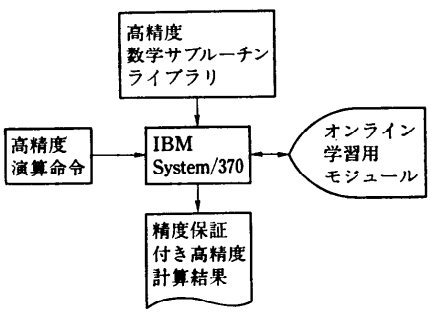

図-1　ACRITH の構成要素

● 高精度数学サブルーチンライブラリは，3レベルの階層で構成されている（図-2）．問題解決ルーチン (Problem Solving Routines)，基本算術演算ルーチン (Basic Arithmetic Routines)，完全精度アキュムレータ関連ルーチン (Full Precision Primitives) の三つで，下位レベルのルーチンが上位のルーチンからの要求を実行するが，どのルーチンも VS FORTRAN から呼び出すことができる．

● 高精度演算命令は，高精度数学サブルーチンライブラリの基礎となるもので，区間演算と高精度演算との最も基本的な命令，すなわち，演算して丸めを行う命令とアキュムレータを対象とする演算命令からなっている．アセンブラ・ルーチンによって書かれているが，IBM 4361 ではハードウェアで提供されている．

● オンライン学習用モジュールは，ACRITH ライブラリを理解し学習するための，またプログラムを書くことなしに ACRITH ルーチンを利用するための，対話型インタフェースを提供する．

以下には，高精度数学サブルーチンライブラリについて説明する．まず上位の問題解決サブルーチンについてそのアルゴリズムをみることにし，それを支えるものとしての下位の二つのルーチン群を紹介する．

2.2 問題解決サブルーチンとそのアルゴリズム

高精度数学サブルーチンライブラリの最上位は問題解決サブルーチンライブラリで，次のものからなる．

- 算術式表現
- 多項式の零点
- 連立1次方程式（密行列）
- 連立1次方程式（疎行列）
- 線形計画法
- 固有値問題
- 連立非線形方程式
- 基本関数

ここでは，連立1次方程式（密行列），固有値問題，算術式表現の三つをとりあげて，そこで採用されているアルゴリズムをみる．

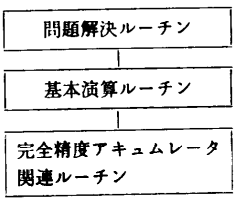

図-2　高精度数学サブルーチンライブラリの構成

以下，次のように記号の約束をする．区間データを表すときは，カギ括弧を用いて，$[x]$, $[A]$ のように，区間の上下限を示すときは $[x]=[x_1, x_2]$ のように書く．

（1） 連立1次方程式(密行列)のアルゴリズム[6),11)]

〈問題〉 連立1次方程式 $Ax=b$ を解く．

〈アルゴリズム〉

a．Aの近似逆行列 R と方程式の近似解 $\tilde{x}$ を求める．

a.1 A の近似逆行列 R を Gauss-Jordan 法で求める．

a.2 近似解 $x_0=Rb$ を初期値として，残差修正法により改良して十分良い近似解 $\tilde{x}$ を求める．

b．保証区間解 $\tilde{x}+[y]$ を求める．

b.1 初期計算：繰返し計算に必要な基本量を求める．

$[z]=R\times(b-A\times\tilde{x})$, $[G]=I-R\times A$,

$[y]=[0]$.

b.2 繰返し計算：区間解ベクトルを改良して包含条件をテストする．

$[x]=[y]\circ\varepsilon$, $[y]=[z]+[G]\times[x]$.

テスト：$[y]\subseteq\mathrm{int}[x]$.

テストが通れば，$\tilde{x}+[y]$ が求める保証区間解である．テストが通らないときは b.2 の最初にもどる．

〈説明〉

• (b.2)での区間改良計算 $[y]=[z]+[G]\times[x]$ は，解の増分に対する残差修正法そのものである．

• (b.2)での $[y]\circ\varepsilon$ は，ε 膨張とよび，区間 $[y]$ の上下限を相対比 ε で拡大したものである．

$[y]\circ\varepsilon=[y]\times[1-\varepsilon, 1+\varepsilon]$.

• (b.2)での $\mathrm{int}[x]$ は，区間 $[x]$ の内部 (x_1, x_2) をさす．

• 繰返し回数は，たとえば 15 回というように設定する．

• いったん包含条件テストを通ったあとで，その保証区間幅の精度が十分でないときは，これに次いで b.2 を繰り返すなり，あるいは得られた保証区間解を用いて点近似解をとり直すなりの方法がある[6)]．

〈基本となること〉

このアルゴリズムでの基本となるところは，(b)での演算が区間演算であることである．これは次のレベルの「基本演算ルーチン」が受けもつ．もう一つ重要なことは，残差修正法による解の改良のためには区間内積演算が高精度で計算されねばならないことであり，またそうすることにより高精度の結果が可能になる．これは最下位の「完全精度アキュムレータ関連ルーチン」が受けもっている．

（2） 固有値問題のアルゴリズム[6),11)]

非線形性が区間改良計算のところに入ってくるが，計算の流れは(1)の連立1次方程式と基本的に同じである．

〈問題〉 固有値問題 $Ax=\lambda x$ を解く (A : n 次正方行列)

〈アルゴリズム〉

a．A の近似固有値 $\tilde{\lambda}$ と対応する近似固有ベクトル $\tilde{x}$ を求める（与える）．

b．保証区間解 $\tilde{\lambda}+[\mu]$, $\tilde{x}+[y]$ を求める．

b.1 初期計算：繰返し計算に必要な基本量を求める．

$G_0=\begin{pmatrix} A-\tilde{\lambda}\times I & -\tilde{x} \\ e_k' & 0 \end{pmatrix}$ の近似逆行列 R を求める．

G_0 の次元は $(n+1)\times(n+1)$ である．$[y]=[0]$, $[\mu]=[0]$ とおく．ここで k は $\tilde{x}$ の非零成分の番号で，たとえば絶対値最大の成分のものをとる．e_k は k 成分のみが1の単位ベクトル，e_k' はその転置行ベクトルである．$\zeta\neq 0$ は定数．

b.2 繰返し計算：区間解ベクトルを改良して包含条件をテストする．

$$\begin{pmatrix}[z]\\ [\nu]\end{pmatrix}=-R\times\begin{pmatrix} A\times\tilde{x}-\tilde{\lambda}\times\tilde{x} \\ e_k'\tilde{x}-\zeta \end{pmatrix}+(I-R\times[G])\begin{pmatrix}[y]\\ [\mu]\end{pmatrix}.$$

ここで，$[G]=\begin{pmatrix} A-\tilde{\lambda}\times I & -\tilde{x}-[y] \\ e_k' & 0 \end{pmatrix}$.

テスト：$\begin{pmatrix}[z]\\ [\nu]\end{pmatrix}\subseteq\mathrm{int}\begin{pmatrix}[y]\\ [\mu]\end{pmatrix}$.

テストが通れば，$\tilde{\lambda}+[\mu]$, $\tilde{x}+[y]$ が保証区間解である．テストが通らないときは，$[y]=[z]$, $[\mu]=[\nu]$ として b.2 の最初にもどる．

（3） 算術式の計算[6),11)]

ここでは簡単な例をとりあげて，原理的なところをみる．

〈問題〉 式：$e=(p\times q-2)/(q+0.1)$ の値を求める．ここで，$p=1.231$ と10進数を文字表現で指定し（内部では分数表現），$q=1.625\mathrm{D}0$ と16進数倍長精度データで指定したとする．

〈アルゴリズム〉 この問題は特殊な連立1次方程式の問題に帰着される．

a．連立1次方程式の形に変形する．

a.1 $e=(c_1/c_2\times c_3-c_4)/(c_3+c_5/c_6)$,

$c_1=1231D0$, $c_2=1.D3$, $c_3=1.625D0$,

$c_4=2.D0$, $c_5=1.D0$, $c_6=1.D1$.

a.2 変数 x を導入して方程式の形に変形する.

$x_1=c_1/c_2$, $x_2=x_1\times c_3$, $x_3=x_2-c_4$,

$x_4=c_5/c_6$, $x_5=c_3+x_4$, $x_6=x_3/x_5$.

最後の式のみ非線形である.

b. 方程式の保証区間解を求める.

この場合は,(1)の連立1次方程式のアルゴリズムを基本として,方程式の特殊性を利用して,簡単な逐次残差修正法を用いる.

〈基本となること〉

式 e を文字表現で与え,またパラメータ p, q に定数を与える仕組みを用意して,上記のように解析,変形して保証区間解を求める機能が必要になる.

2.3 高精度演算ルーチンと完全精度アキュムレータ関連ルーチン

(1) 高精度演算ルーチン

このルーチンは大きく変換ルーチンと演算ルーチンに大別される.データとしては,実数と複素数,点データと区間データ,単精度と倍精度のそれぞれがあるので,以下のルーチンにはそれらの組合せに応じたものが用意されている.

- 変換:外部10進表現と内部16進表現の間の変換を行う.変換は最大精度で行われる.
- スカラ演算:二つのスカラ・データ間の四則演算を行う.この際有向丸めを行う.
- ベクトル演算:二つのベクトル・データの間の加減算,ベクトルのスカラによる乗算とスカラによる除算を,有向丸めとともに行う.
- 内積演算:二つのベクトルの内積をアキュムレータ内に完全精度で求め,有向丸めにより最大精度で結果を出す.
- 行列乗算:二つの行列の積を完全精度で求め,有向丸めにより最大精度で結果を出す.

ここで最大精度 (maximum accuracy) とは,有効桁の最後の桁の1以内の精度を保証していることをさす.これに対して完全精度 (full accuracy) は,次の(2)で述べるように,アキュムレータという機構を用いて,内部浮動小数点数の積和を厳密に誤差なしに保持することをさす.有向丸めには4種あり,「ゼロに向かって」(to zero),「最も近いものへ」(nearest),「上方に」(upwards),「下方に」(downwards),それぞれ状況に応じて丸めることができる.

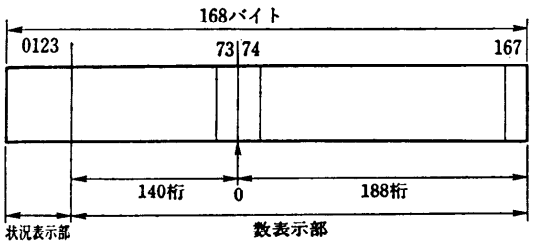

図-3 アキュムレータのレイアウト

(2) 完全精度アキュムレータ関連ルーチン

このルーチンは高精度計算を可能とするためのアキュムレータ関連のものである.一つのアキュムレータは,図-3 に示すように168バイトで構成されている.

168バイトのうちの頭4バイトが状況表示部 (status area) で,数の符号(1バイト目第1ビット),数値が占めている上限の位置(第3バイト),下限の位置(第4バイト)をそれぞれ示す.第2バイトは使用していない.残りの 164 バイトが数表示部で,74 バイト目と 75 バイト目に小数点がある.16進で整数部は 140 桁,小数部は 188 桁ある.IBM S/370 の場合は,指数は63から −64 までであるので,積和は完全精度で十分保持できる.

アキュムレータ関連ルーチンは次の三つに分かれる.

- アキュムレータの定義:アキュムレータを記憶域に設定する.実数用アキュムレータと複素数用アキュムレータがある.また区間データに対しては,上下限用それぞれのアキュムレータを設定する.
- アキュムレータ演算:アキュムレータのクリア,アキュムレータへの加算,アキュムレータからの減算,二つのベクトルの内積の加算などを行う.
- アキュムレータからのストア操作:アキュムレータから有向丸めにより通常の浮動小数点表現のものにストアする.

3. ACRITH による計算例

この章では,ACRITH による簡単な数値計算例をいくつかあげて,その機能を具体的にみる.また FORTRAN-SC[7),8)] による例をあげて,この上位のインタフェースによって簡明にプログラムが書けることを示す.なお,FORTRAN-SC の特徴は付録に簡単にまとめている.

3.1 ACRITH を直接用いる計算例

（1） 連立1次方程式・ケース1[10),12)]

a．問題：次の連立1次方程式の解を求める．

$6491912x-159018721y=1,$

$41869520.5x-102558961y=0.$

b．結果

●結果 1：通常のガウス消去法で求めた結果

$x=0.8481464640570600\mathrm{D}+08$

$y=0.3462543440142650\mathrm{D}+08$

●結果 2：連立1次方程式のサブルーチン DLIN (2.2(1)のアルゴリズムを倍長精度・点データ入力の場合に適用したもの) を用いた結果．なお，結果の区間出力には変換サブルーチン DIOUT を用いる．また，区間の出力表示は開き括弧で示す（以下同様）．

$x=$(0.20511792199999999D9,
0.20511792200000001D9)

$y=$(0.83739040999999996D8,
0.83739041000000004D8)

真の解は，$x=205117922$, $y=83739041$ である．

（2） 数値微分[10),12)]

a．問題：次の関数 $f(t)$ について，$f''(1)$ の値を中心差分を用いて求める．

$f(t)=(4970t-4923)/(4970t^2-9799t+4830),$

$f''(x)\sim(f(x+h)-2f(x)+f(x-h))/h^2.$

b．結果：通常の FORTRAN によるものと ACRITH によるものを比較する．真の値 $f''(1)=94$ である．$f''(x)$ を求める差分式を文字表現で与え (FVAL 使用)，この式の中に現れるパラメータ x,h に定数を文字で与え (FDASG 使用)，この式の計算を行う．h は 10^{-4} から 10^{-10} まで動かした．

ORDINARY COMPUTATION RESULTS:

H=10**	
−4	70.79399895284251
−5	94.40949832618242
−6	185.16764838527242
−7	−3026.55886273188025
−8	30695.46291576426390
−9	−447642.04824571963400
−10	−90949455.12690887970000

ACRITH RESULTS:

H=10**	
−4	(0.7078819087920 20D+02, 0.7078819087920 22D+02)
−5	(0.9376790475465 09D+02, 0.9376790475465 10D+02)
−6	(0.9399767904983 09D+02, 0.9399767904983 10D+02)
−7	(0.9399997679049 85D+02, 0.9399997679049 86D+02)
−8	(0.9399999976790 49D+02, 0.9399999976790 50D+02)
−9	(0.9399999999767 90D+02, 0.9399999999767 91D+02)
−10	(0.9399999999997 67D+02, 0.9399999999997 68D+02)

（3） ロジスティック型2次式の繰返し計算[8)]

a．問題：$x_{n+1}=4\lambda x_n(1-x_n)$, $0<\lambda\leqq 1.$

初期値 $x_0=0.3$ として，$\{x_n\}$ の挙動をみる．

b．結果：

$\lambda=0.8$（2点サイクル）と $\lambda=0.954$（カオス的領域）の2ケースの結果を示す．通常の場合と，問題を $x_0, x_1, \cdots, x_{N-1}$ に対する非線形方程式と考えて ACRITH のサブルーチン DNLSS を使用した場合を比較する．

●ケース 1：$\lambda=0.8$

ORDINARY COMPUTATION RESULTS:

X030	=	0.79945547633013
X031	=	0.51304453662687
X032	=	0.79945548820541
X033	=	0.51304451386771
X034	=	0.79945549010546
X035	=	0.51304451022624
X036	=	0.79945549040946
X037	=	0.51304450964361
X038	=	0.79945549045810
X039	=	0.51304450955039
X040	=	0.79945549046589

ACRITH RESULTS:

X030	=	(0.7994554763301D+00, 0.7994554763302D+00)
X031	=	(0.5130445366268D+00, 0.5130445366269D+00)
X032	=	(0.7994554882054D+00, 0.7994554882055D+00)
X033	=	(0.5130445138677D+00, 0.5130445138678D+00)
X034	=	(0.7994554901054D+00, 0.7994554901055D+00)
X035	=	(0.5130445102262D+00, 0.5130445102263D+00)
X036	=	(0.7994554904094D+00, 0.7994554904095D+00)
X037	=	(0.5130445096436D+00, 0.5130445096437D+00)
X038	=	(0.7994554904581D+00, 0.7994554904582D+00)
X039	=	(0.5130445095503D+00, 0.5130445095504D+00)
X040	=	(0.7994554904658D+00, 0.7994554904659D+00)

●ケース 2：$\lambda=0.954$

ORDINARY COMPUTATION RESULTS:

X000	=	0.30000000000000
X010	=	0.86493814920877
X020	=	0.26148857153031
X030	=	0.90959721004922
X040	=	0.28402609920251
X050	=	0.59559504830118
X060	=	0.94801669229650
X070	=	0.92146470408723
X080	=	0.58099365857730

ACRITH RESULTS:

X000	=	(0.2999999999999D+00, 0.3000000000001D+00)
X010	=	(0.8649381492087D+00, 0.8649381492088D+00)
X020	=	(0.2614885715285D+00, 0.2614885715286D+00)
X030	=	(0.9095972104041D+00, 0.9095972104042D+00)
X040	=	(0.2840260063537D+00, 0.2840260063538D+00)
X050	=	(0.5955912948946D+00, 0.5955912948947D+00)
X060	=	(0.9480474416038D+00, 0.9480474416039D+00)
X070	=	(0.9290968225709D+00, 0.9290968225710D+00)
X080	=	(0.9498040714091D+00, 0.9498040714092D+00)

c. DNLSS の使用は次のとおりである（文献 8）から引用，一部変更）.

```
CALL DNLSS (N, FKT, VNAME,
            VSTART, 12, RESL, RESU,
            IR, IFKT, IPOS)
.
.
.
DO 10 I=1, N, IC
  CALL DIOUT (RESL (I), RESU (I),
              13, OUT)
  WRITE (*, *) VNAME(I), ' = ' ,OUT
10 CONTINUE
```

〈説明〉 DNLSS の引数

1. N : $x_0, x_1, \cdots, x_{N-1}$ を求める.
2. FKT は文字列配列で，非線形方程式の左辺を表す（右辺はゼロ）.

```
FKT(1) : 'X000-X0'
FKT(2) : 'X001-4*LAMBDA*X000
                    *(1-X000)'
FKT(3) : 'X002-4*LAMBDA*X001
                    *(1-X001)'
```

のように作成する.

3. VNAME は式に現れる変数である.

```
VNAME(1) : 'X000', VNAME(2) : 'X001'
```

のように作成する.

4. 式に現れるパラメータ X0(=x_0) や LAMBDA (=λ) は，定数文字列で与える.
5. VSTART は VNAME の初期値を与える配列である.
6. 有効 12 桁まで正しい解を求める.
7. RESL, RESU に保証区間解の下限，上限が求

```
      PROGRAM INEWT
      DOUBLE INTERVAL X, Y, L, M, MIDPNT, MIDDLE, F, DERIV          ①
      LOGICAL CRITER
      EXTERNAL F, DERIV, MIDPNT, CRITER
  111 WRITE(*,*) 'PLEASE ENTER STARTING INTERVAL'
      READ(*,*,END=999) Y
      IF( CRITER(Y) ) THEN
       REPEAT
        X=Y
        MIDDLE=MIDPNT(X)
        Y=( MIDDLE - F(MIDDLE)/DERIV(X) ) .IS. X                    ②, ④
        WRITE(*,*) Y
       UNTIL (X .EQ. Y)                                             ②
      ELSE
       WRITE(*,*) ' CRITERION NOT SATISFIED '
      END IF
  999 END
      FUNCTION F(X)
      DOUBLE INTERVAL F, X                                          ①
      F = SQR(X)*( SQR(X)/3 + SQRT((<2>))*SIN(X) ) - SQRT((<3>))/19 ②, ③
      END
      FUNCTION DERIV(X)
      DOUBLE INTERVAL DERIV, X                                      ①
      DERIV=X*( SQR(X)*4/3 + SQRT((<2>))*(SIN(X)*2 + X*COS(X)) )    ②, ③
      END
      FUNCTION MIDPNT (X)
      DOUBLE INTERVAL MIDPNT, X                                     ①
      MIDPNT= IVAL ( INF(X) + 0.5*(SUP(X) - INF(X)) )               ②, ③
      END
      FUNCTION CRITER (X)
      DOUBLE INTERVAL X, F, DERIV                                   ①
      LOGICAL CRITER
      EXTERNAL F, DERIV
      CRITER= (0. .IN. F(X) ) .AND. .NOT. (0. .IN. DERIV(X) )       ④
      END
```

図-4 ニュートン法による求根プログラム

まる. (IR, IFKT, IPOS の説明は省略)

なお, パラメータを区間データで与えることのできるサブルーチン DCQSP5 が, 研究目的用にソースプログラムで提供されている.

3.2 FORTRAN-SC から ACRITH を用いる計算例

FOTRAN-SC利用の2例をあげる. 直接 ACRITH で書くと相当はんさになる. たとえば次の(1)の場合には, ACRITH では約70ステートメントになる.

(1) 非線形方程式[3),8),15)]

a. 問題: 次の方程式の [0, 1] の間の根をニュートン法を用いて求める.

$$f(x)=x^2(x^2/3+\sqrt{2}\sin x)-\sqrt{3}/19=0$$

b. 結果 (保証区間解):

(0.392379328…D+00, 0.392379576…D+00)

c. プログラム: FORTRAN-SC の場合は, **図-4** に示すように自然な形でアルゴリズムを記述することができる (文献 8) から引用).

〈説明〉

① データタイプに DOUBLE INTERVAL が使える.

② 区間データに普通の演算記号がそのまま使える (+ − ＊ ／, .EQ.).

③ 区間データに関する組み込み関数が用意されている.

SQR, SQRT, SIN, COS

INF, SUP: 区間の下限, 下限

IVAL: 区間データに変換

④ 区間データに対する新しい算術演算子 .IS. や, 新しい関係演算子 .IN. が用意されている.

.IS.: 二つの区間の共通集合

.IN.: 点データが区間に属しているか.

(2) 自己検証的 SOR 法[13),14)]

a. 問題: SOR 法に区間演算を用いて保証区間解を求める. 方程式を $Ax=b$ とする.

b. アルゴリズム:

b.1 適切な加速因子 ω による点 SOR 法により近似解 $\tilde{x}$ を求める.

b.2 $[\tilde{x}]$ を初期値として, 加速因子 ω による区間 SOR 法により保証区間解 $[x]$ を求める. すなわち, 各要素ごとに区間 Gauss-Seidel 法による改良区間 $[z]$ を求め, 前の区間に含まれているかどうかを検証する. 含まれていない成分があるときは, 区間中点 $m[x]$, $m[z]$ に対する SOR 法を適用して中点 my を求め, それに $[z]$ の区間幅 $d[z]$ をつけたものを改良区間とする. **図-5** に疑似コードでアルゴリズムを記述する.

```
[x]=[x̃, x̃]
do k=1, kmax
  incl=.TRUE.
  do i=1, n
    [z]=(b_i − Σ_{j≠i} a_ij×[x_j])/a_ii                ①
    if([z]⊈[x]) then                                   ②
      incl=.FALSE.
      my=(1−ω)×m[x_i]+ω×m[z]
      [y]=[my−d[z]/2, my+d[z]/2]                       ③
      if([y]⊆[x_i]) then                               ②
        [x_i]=[y]
      else
        [x_i]=[y]×[1−ε, 1+ε]                           ③, ④
      endif
    endif
  enddo(i)
  if(incl) then
    goto exit
  endif
enddo(k)
exit: if(incl) then
    return('included', [x̃]=[x])
  else
    return('not included')
  endif
```

図-5 自己検証的 SOR アルゴリズム

〈説明〉

① Σ の部分の積和に対して区間ベクトルの内積演算記号＊がそのまま使える.

② 区間データに対する関係演算子 .SB. (subset), または .SP. (superset) が使える.

③ IVAL により区間データに変換できる.

④ 区間データの乗算に普通の乗算記号をそのまま使える.

4. おわりに

本稿では, 精度保証付き数値計算ソフトウェアの一つである IBM の ACRITH について, その機能と構成, 簡単な数値計算例を紹介した. また, ACRITH を土台にした高水準インタフェースの FORTRAN-SC についても, その使用例をあげた. こういったソフトウェアによって, 数値計算の品質保証を現実のものにすることができるようになったことは大きな前進である.

とはいえ, 今後考えていくべき課題もある. 一つは実用の問題である. 実際に使用されている範囲がまだ広くない. とくに「生産的」数値計算の現場からはまだかけ離れたところにあるといっても過言ではな

い．たしかに，区間演算には点演算—通常の演算—の数倍の計算時間がかかるし，精度保証付き計算アルゴリズムのためにはそれ相当の時間がかかるから，あらゆる計算に適用するのは非現実的であろう．しかし，きびしい安全性が要求される問題，悪条件性が予想される問題に対してはそれだけの計算時間をかける値打が十分あるし，またそうしなくてはならないはずである．どのようにして実用にしていくかが問われている．またそのためには，プログラムを最初からFORTRAN-SCのような高水準言語で書くことが必要だろう．また実際の計算では，自己検証的SOR法のように，よい近似値を求めたうえで若干の追加計算により保証区間解を求める方法が望ましい．

もう一つの問題は，いわゆるスーパ・コンピューティング，ベクトル計算との関連である．ここでの要となる演算もまた累和演算である．ベクトル・プロセッサでのパイプライン技術を高精度演算に拡張したコンピュータが必要だが，そのためのアーキテクチャも研究が進められてきている[16]．数式処理技術との組合せもまた必要である．

これらのいろいろな課題を解決しつつ，精度保証付き計算が数値計算の原則となることが期待されている．

参考文献

1) Wilkinson, J. H.: Rounding Errors in Algebraic Processes, Her Majesty's Stationary Office, Prentice Hall, London (1963).
2) Knuth, D. E.: The Art of Computer Programming, Vol. 2, 2nd ed., p. 213 (1981).
3) Alefeld, G. and Herzberger, J.: Introduction to Interval Computations, Academic Press (1983). (revised and expanded English version of the text, 1974)
4) Kulisch, U. W. and Miranker, W. L.: Computer Arithmetic in Theory and Practice, Academic Press (1981).
5) ACRITH General Information Manual, Publication Number GC 33-6163, IBM (1986).
6) IBM High-Accuracy Arithmetic Subroutine Library: Program Description and User's Guide, Publication Number SC 33-6164, IBM (1986).
7) Kulisch, U. and Middel, J.: FORTRAN-SC (A FORTRAN Extension for Engineering/Scientific Computation with Access to ACRITH) Language Reference and User's Guide, IBM (1989).
8) Kulisch, U. and Middel, J.: FORTRAN-SC (A FORTRAN Extension for Engineering/Scientific Computation) General Information Notes and Sample Programs, IBM (1989).
9) Kulisch, U. W. and Miranker, W. L. (eds.): A New Approach to Scientific Computation, Academic Press (1983).
10) Kulisch, U. W. and Miranker, W. L.: The Arithmetic of the Digital Computer: A New Approach, SIAM Review, Vol. 28, No. 1, pp. 1-40 (1986).
11) Rump, S. M.: Solving Algebraic Problems with High Accuracy, in Ref. 9).
12) 奥田 晃，棚町芳弘：ACRITH（高精度演算サブルーチン・ライブラリー）—その概要と数値計算例—，ワークショップ「科学技術計算の動向」資料集，愛媛大学 (1986).
13) 棚町芳弘，奥田 晃：Some Experiments on New ACRITH—Self-Validating SOR Algorithm—，数理科学講究録 673，自己検証的算法とその応用，pp. 22-39 (1988).
14) 棚町芳弘，奥田 晃：自己検証的 SOR について，日本数学会春季応用数学分科会講演アブストラクト，pp. 43-46 (1989).
15) Metzger, M.: FORTRAN-SC A FORTRAN Extension for Engineering/Scientific Computation with Access to ACRITH Demonstration of the Compiler and Sample Programs, in Moore, R.E. ed., Reliability in Computing, The Role of Interval Methods in Scientific Computing, Academic Press (1988).
16) Kirchner, R. and Kulisch, U.: Arithmetic for Vector Processors, in the same book as Ref. 15).

付録 FORTRAN-SC の特徴

1. 区間データを定義し，区間演算を直接指定できる．

2. 動的アレイ機能：プログラム実行時にアレイの記憶域を割り振ったり解放したりして，記憶域を経済的に使用できる．

3. 内積表現：次のような表記により点データを対象とする内積計算を厳密に行う．ここで s, v, m は，それぞれスカラ，ベクトル，行列を表す．

$s1+s2*s3-v1*v2$（結果はスカラ）

$v1-m1*v2+s1*v3$（結果はベクトル）

$m1+m2*m3-s1*m4$（結果は行列）

この表現のなかで累和記号 Σ に対応する記号SUMが使用できる．結果に対する有向丸め記号も用意されている．

4. アレイ値FUNCTION：関数値がアレイであるFUNCTIONを定義できる．

5. ユーザ定義オペレータ：一つまたは二つの引数をもつ(ユーザ定義)関数を単項または2項オペレータとして定義できる．

（平成2年4月2日受付）

訂　　正

本誌**第31巻9号**（1990）pp. 1220〜1227 に掲載されました「大型計算機向き汎用精度保証付き数値計算ソフトウェア」の著者奥田晃氏の申し出により，pp. 1224 の左段を以下のとおり訂正します．

	(誤)	(正)
(i)	4行目	
	$6491912\,x-159018721\,y=1$	$64919121\,x-159018721\,y=1$
(ii)	7〜9行目	
	・結果1：通常のガウス消去法で求めた結果	・結果1：通常のガウス消去法のプログラムによる結果
	$x=0.8481464640570600\,\mathrm{D}+08$	$x=0.14135774\,\mathrm{D}\,09$
	$y=0.3462543440142650\,\mathrm{D}+08$	$y=0.57709057\,\mathrm{D}\,08$