〈報文〉

多倍精度の４則演算について

千葉職業能力開発短期大学校千葉校　　加　部　通　明

Four Arithmetic Operations for Multiple Precision

Michiaki KABE

要約　計算機で有理数を係数にする連立１次方程式の厳密解を求めたり、ベルヌーイ数を計算したりするときには、計算途中で有理数の四則演算を行わなければならない。扱う数の桁数が小さい場合は倍精度程度の桁でよいが、桁数が大きい場合には多倍精度による計算が必要になる。

本稿はそうした桁数の大きい計算問題に対して、パソコン上でも手軽に計算できる多倍精度による四則演算、特に、除算のアルゴリズムと、それを応用したプログラム及びその応用結果について報告する。

Ⅰ　はじめに

数値解析の授業で、簡単な連立１次方程式を計算機で求める課題を出した。手算で解いてみればきれいな整数解になるのが、プログラムを作成して計算機にかけて出てきたものが小数点以下に、９や０がいくつか続いた後、最後に１や８が出てきたりするのを見て、学生達は不思議な顔をする。自分達がつくったプログラムなので少し考えれば納得するのだが、精確無比なコンピュータが自分達以下の精度の解を出したということが不満なのである。

連立１次方程式の解法の中で一番簡単で精度が良いのがガウスの消去法であるが、そこには必ず割算が伴う。この係数の消去の過程で誤差が生じるのである。では、誤差が生じないようにするにはどうしたらよいか、逆に学生達に問いかけてみると、彼らは途中の計算過程で分数のままにしておけばよいことを指摘する。

そこで、分数が扱えるようにプログラムを作成する課題を出すとなかなかできない。有理数での四則演算は常に約分することを考えねばならない。そうしないと分子と分母の桁数が増えて桁上がりが頻繁に発生し、なおかつ、余分な計算までするので非効率的である。ユークリッドの互除法を用いて約分するのである。この操作が結構面倒なのである。しかし、この約分を行っても３桁程度の係数で５元までの方程式ぐらいまでしか解けない。もっと大きな連立方程式を解くには多倍精度による演算アルゴリズムを開発する必要がある。

既存の言語や数式処理プログラムの中でも、例えば、UBSICは2,600桁[1]、REDUCEはメモリの許す限り[2]の有理数計算ができるものがある。しかし、REDUCEを実際使用してみると、記憶保護が働き200桁ぐらいの計算しかできない。あえてここで本稿を著した理由は、数式処理プログラムを作成するのに必要な多倍精度による四則演算の提案するアルゴリズムが簡単で、手軽なパソコン上でプログラムが作成でき、しかも、使用する計算機のメモリの許す限りの桁数を扱うことができるからである。

本稿は特にその応用として、連立１次方程式の厳密解の計算やベルヌーイ数について述べる。理論的には、パソコンレベルで約25,000桁の四則計算が可能であり、実際、8000!まで計算した。さらにREDUCEと本プログラムの計算量の限界比較を行った。

Ⅱ　多倍精度の四則演算

１　有理数を扱う多倍精度について

多倍精度計算の例として円周率πや$\sqrt{2}$を何億桁、何百万桁計算したという話はよく聞く。除数の桁数が小さくて被除数の桁数が大きい場合は比較的簡単なプ

論文受付日　1992.12.22

ログラムですむ。ところが、除数、被除数ともに桁数が大きい計算では簡単な割算のアルゴリズムは存在しない。多項式の因数分解に相当する試行錯誤がいる。但し、必ず除算ができなければならないが。

計算機の倍精度は機種によって異なるが、パソコンのBASIC言語だと通常16桁である。ここでは、簡便化して6桁を倍精度として以下話を進めよう。

2　変数の取り扱いについて

大きな桁数を扱う場合、変数は配列を使用する。例えば変数Aが30桁必要であれば、6×5より配列

A(1)；A(2)；A(3)；A(4)；A(5)

が1つの変数を表す。但し、実際には四則演算によって桁上がりが生じるので6桁は使用できない。加算及び乗算では常にこの桁上がりに注意しなければならない。

配列の次元を大きくすればそれだけ大きな桁数の四則演算ができるが、計算機のメモリの大きさにより桁数が制限される。ここでは、当初メモリの制約は考えずに話を始める。

そこで変数Aの配列の次元をNとすればAの取る値は左側が大きな位として

A = A(1)；A(2)；…；A(N)　　(1)

で表現される。これを数学的に書けば、$E=10^6$と置いて

$A = A(1)\times E^{N-1}+A(2)\times E^{N-2}+A(N-1)\times E+\cdots+A(N)$

となる。

(1)の形の数Aで、各A(i)の符号が一致していて、かつ｜A(i)｜<Eとなっているとき、Aを標準形と呼ぶことにしよう。

3　加算と減算

1）加算は計算によって桁上がりがあっても1桁である。よって1配列で使用する桁数は5桁だから、$E=10^5$となる。

実際の計算A＋Bは次のように行う。但し、A, B, C, は(1)のように表されているものとする。

C = A＋B = A(1)＋B(1)；A(2)＋B(2)；…；A(N)＋B(N)

今仮にi列でC(i)≧Eとなれば、i-1列に1を加え、i列からE引かなければならない。

2）減算については、桁上がりがないので1配列6桁のままでよいのだが、四則演算を行うとき、加算と減算の桁上がりを変えて計算するのは煩雑であるので、小さい桁数の5桁を採用する。

また、減算には少しやっかいなところがある。例をあげて考えよう。今、N=4として

A=0；4590；2340；98030　B=0；15244；97；38089

なるとき、A-B=Cを計算する。これを筆記で計算すると

A-B=45900234098030-152440009738089

=-106539775640059

となる。

ところが計算機では

C = A-B = A(1)-B(1)；A(2)-B(2)；A(3)-B(3)；A(4)-B(4)

でもって計算するので

C=0；-10654；2243；59941

となり実際と異なる。これを解消するために、負の数を表す場合には各配列を全て負の数にすればよい。

つまり

-106539775640059=0；-10653；-97756；-40059

とする。逆に、上の数値A, Bにたいして、B-Aを計算すると

B-A=0；10654；-2243；-59941

となるのでB-Aが正より、各配列を正にして

B-A=0；10653；97756；40059

とすればよい。

[規則]

これを一般のNについて行う場合は、0以外の数値が最初に現れる配列の符号がA-Bの正負を決定するのでその符号に各配列を合わせる。

以上の内容をプログラムで表すと次のようになる。

```
      SIGN=1
      DO    10   I=N,1,-1
                 C(I)=A(I)-B(I)
                 IF        (C(I).LT.0)    THEN
                           SIGN=-1
                 END IF
10    CONTINUE
      IF         (SIGN.EQ.-1)    THEN
                 DO   20   I=N,2,-1
                           IF        (C(I).GT.0)    THEN
                                     C(I)=C(I)-E
                                     C(I-1)=C(I-1)+1
                           END IF
20                CONTINUE
      ELSE
                 DO   30   I=N,2,-1
                           IF        (C(I).LT.0)    THEN
                                     C(I)=C(I)+E
                                     C(I-1)=C(I-1)-1
                           END IF
30                CONTINUE
      END IF
```

図1　減算プログラム

4 乗算

1）先ずA，Bが共に正の時を考える。

加減算による倍精度6桁の変数は5桁使用された。この5桁で最大の数は99999だからそれらを掛け合わせれば、

99999×99999=9999800001

となり、10桁の数となる。従って、掛けた結果が6桁になるには元の数を3桁にしなければならない。更に、A，BはN次元の変数であるから

C=A×B=A(1)；A(2)；…；A(N-1)；A(N)

×) B(1)；B(2)；…；B(N-1)；B(N)

A(1)B(N)；…；A(N-1)B(N)；A(N)B(N)

A(1)B(N-1)；…；A(N-1)B(N-1)；A(N)B(N-1)

……

+) A(1)B(1)；…；A(N-1)B(1)；A(N)B(1)

C(1)；C(2)；　　……　　；C(2N)

なる計算を実行すればある配列C(i)では最大N個の和を計算する可能性がでてくる。その時の桁上がりは

999×999×N=998001N

より最大$\log_{10}(9.99N)$でN=100の時2桁増える。

よって、最低1桁ぐらいは余裕をもたせる必要があるので乗算では1配列2桁となる。

そこで、各配列別に積A(i)B(j)の和を取って、その後標準化する。

ところで、N次元配列の数を掛け合わせればその結果2N次元配列になるのでCは

C=C(1)；C(2)；…；C(N)；C(N+1)；…；C(2N)

としておかなければならない。しかし、数によって次元が異なると桁数を把握するのに面倒であるから、変数の次元は全て同じにして、掛け合わせるもの同志の次元をNの半分にすると乗算のアルゴリズムが簡単ですむ。

```
      K=0
      DO   10   J=N,JS,-1
                DO   20   I=N,IS,-1
                          W(I-K)=A(I)*B(J)
20              CONTINUE
                DO   30   I=N,IS-K,-1
                          T=IDINT(W(I)/E)
                          W(I)=W(I)-T*E
                          W(I-1)=W(I-1)+T
30              CONTINUE
                DO   40   I=N,1,-1
                          C(I)=C(I)+W(i)
40              CONTINUE
10    CONTINUE
```

図2　乗算プログラム

2）A，Bのうち少なくても1つが負の数である時は、負の数の絶対値を取って共に正にして、以下(1)のように行えばよい。その後、計算結果にA，Bの符号を各配列に掛ければよい。

3）乗算の効率的な計算方法

配列の次元が大きいにも拘わらず、実際の数の桁数が小さい場合には、数の桁数を表す引数を用意して、その引数の範囲以内で計算させれば余分な0同志のかけ算をせずにすむ。除法の場合には必ずこの桁数を表す引数がいる。具体的には、それを配列の添え字で代用している。

更に、数万桁以上の乗算では離散的フーリエ変換を使用し高速化をはかる必要があるが、その実用となるプログラムの作成はかなり難しいようである。[3]

5 除算

1）除算のアルゴリズム

A，Bを正の整数とするとき、除算A÷Bは次の除法の原理によって求められる。

$$A=B*Q+R \qquad (0\leqq R<B) \qquad (2)$$

但し、Qは商、Rは余りである。

(2)を利用すれば、メモリの許す限り何桁の数でも除算可能である。今、被除数A、除数Bが次のように表されているとする。

$A=A_1；A_2；\cdots；A_N$　　$A_i=0$　$(1\leqq i<I)$，$A_I\neq 0$

$B=B_1；B_2；\cdots；B_n$　　$B_j=0$　$(1\leqq j<J)$，$B_J\neq 0$

この時もし、A<BならばQ=0，R=Aで除算する必要ないのでA≧Bと仮定する。

これからA÷BをA/Bで表す。今、倍精度が6桁で各配列の成分A_i，B_jが2桁であることを思い出そう。

除算を効率よく行うためには商を効率よく見つけることが肝要である。桁あふれしないようにできるだけ、被除数、除数、商が大きな桁数となるようにとる。

除算のアルゴリズムを示すと次のようになる。[3]

①　$A'=(A_I*E+A_{I+1})*E+A_{I+2}$

$B'=B_J*E+B_{J+1}$

とおき、A'/B'を計算する。この商をQ'として、

$$A-(B*Q')*E^L \qquad (0\leqq L\leqq J-1) \quad (3)$$

を乗算により計算する。

②(3)の値が非負ならば、最初のA/Bの商Q'が確定する。また、もし、(3)の値が負ならばQ'から順次1を引いて(3)の値が非負になるまで繰り返して

行う。
③確定した(3)を改めてAとおく。つまり、

$$A = A-(B * Q') * E^{L}$$

そして、①，②をA<Bになるまで繰り返す。

①，②の途中計算で桁あふれに注意しなければならない。

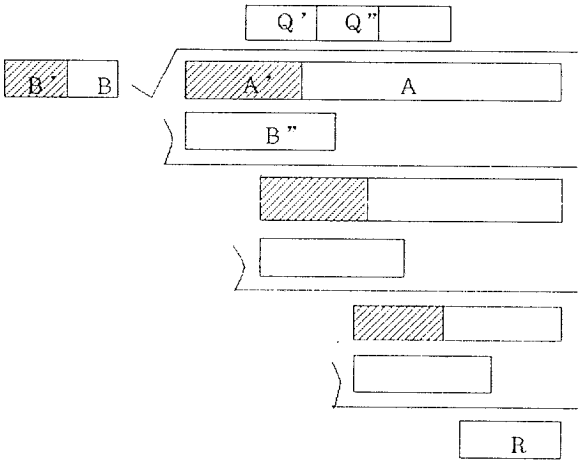

図3　除算

2）例

次の簡単な例を計算しよう。

10967234÷89504

但し、ここでは商と余りを求める。

今、N=5として、A, Bの桁数を表す添え字をそれぞれIS, JSとすれば、乗算のところでわかったように、1配列の桁数は2だから

A=0；10；96；72；34, IS=2　　B=0；0；8；95；04, JS=3

とかける。そこで、除法の基本的な算法は通常のわり算と同じようにして行えばよい。但し、注意しなければならないのは、A<Bであれば計算せずに終わるということである。このことは途中の計算過程でたびたび使用する。

即ち

A(IS)/B(JS)

の商を計算し、それをC(1)とおき

もし

$$A-B * C(1) * E^{IS-JS}<0$$

であれば

C(1)=C(1)-1

として、同じように、$A-B * C(1) * E^{IS-JS}$の計算をしてそれが0以上になるまで上の過程を繰り返す。今の場合、C(1)=1だから具体的には

$A(2) * E^2+A(3) * E+A(4)-C(1) * (B(3) * E^2+B(4) * E+B(5))=109672-89504=20168>0$

となる。実際のプログラムでは次のようになる。

最後に、計算効率を上げるためにできるだけ倍精度の桁数いっぱいを使用する工夫が必要である。

```
      L=1
      JK=N-1
      KEYD=1
      DO   10   I=1,N
                WA(I)=A(I)
   10 CONTINUE
  100 IF   (WA(JK+1).EQ.0) AND (JK.LT.IS))   THEN
                JK=JK+1
                L=L+1
                GO TO 100
      END IF
      KC=0
      IF   (IS.LT.N)   THEN
                T=WA(IS)*E+WA(IS+1)
                KC=1
      ELSE
                T=WA(IS)
      END IF
      IF   (WA(IS).LT.B(JS))   THEN
                S=B(JS)
      ELSE
                KC=0
                IF   (JS.LT.N)   THEN
                          S=B(JS)*E+B(JS+1)
                ELSE
                          T=WA(IS)
                          S=B(JS)
                END IF
      END IF
      Q=IDINT(T/S)
      IF   ((Q.EQ.0) AND (IS.LT.JS))   THEN
                Q=IDINT((T*E+A(IS+2))/S)
                KC=2
      END IF
  200 DO   15   I=1,N
                D(I)=0
                W(I)=0
   15 CONTINUE
      DO   20   I=JS,N
                W(I)=Q*B(I)
   20 CONTINUE
      DO   25   I=N,JS,-1
                U=IDINT(W(I)/E)
                W(I)=W(I)-U*E
                W(I-1)=W(I-1)+U
   25 CONTINUE
      IF   (W(JS-1).NE.0)   THEN   KC=1
      K=JS-IS
      KEYC=0
      DO   30   J=JS-KC,N
                D(J-K+KC)=WA(J-K+KC)-W(J)
                IF    (D(J-K+KC).NE.0)   THEN
                          KEYC=1
                END IF
   30 CONTINUE
      JK=N-K+KC
      DO   35   I=JK+1,N
                D(I)=WA(I)
   35 CONTINUE
      DO   40   I=N,IS+1,-1
                IF   (D(I).LT.0)    THEN
                          D(I)=D(I)+E
                          D(I-1)=D(I-1)-1
                END IF
   40 CONTINUE
      KEYA=0
      DO   45   I=N,IS,-1
```

```
            IF   (D(I).LT.0)   THEN
                     KEYA=-1
            ELSE
                     IF   (D(I).GT.0)   THEN
                              KEYA=1
                     END IF
            END IF
 45 CONTINUE
    IF   (KEYA.EQ.-1)   THEN
            Q=Q-1
            GO TO 200
    ELSE
         IF   ((Q.EQ.0) AND (IS.LT.JS)) THEN
              Q=B-1
              KC=1
              GO TO 200
         ELSE
              IF   (KEYD.EQ.0)   THEN
                     L=L+KC+N-JS
              END IF
              KEYD=1
              IF    (KEYC.EQ.1)   THEN
                     KEYD=0
              END IF
              C(L)=Q
              L=L+1
         END IF
    END IF
    DO   50   I=1,N
            WA(I)=0
 50 CONTINUE
    WS=N
    DO   55   I=N,IS,-1
            WA(I)=D(I)
            IF   (WA(I).NE.0)   THEN   WS=I
 55 CONTINUE
    IS=WS
    KEYB=0
    IF   (KEYA.EQ.0)   THEN   GO TO   300
    IF   (IS.GT.JS)   THEN
            GO TO   300
    ELSE
         IF   (IS.EQ.JS)   THEN
              DO   60   I=N,IS,-1
                   W=WA(I)-B(I)
                   IF   (W.LT.0)   THEN
                            KEYB=-1
                   ELSE
                        IF   (W.GT.0)   THEN
                            KEYB=1
                        END IF
                   END IF
 60            CONTINUE
         END IF
    END IF
    IF   (KEYB.NE.-1)   THEN
            GO TO 100
    END IF
300 K=1
    L=L+N-JK
    DO   65   I=N,N+2-1,-1
            Q(I)=C(L-K)
            K=K+1
 65 CONTINUE
```

図4　除算プログラム

III　基本演算モジュール

基本的な演算はモジュールとして形式を統一しておくと便利である。それは、いろいろな数式処理プログラムの作成に際し、モジュールを組み立てることによってある程度の演算が構成され、プログラムの生産性向上に役立つからである。

1　基本的な四則演算のモジュール

英文字は正の整数を表す。

①積 A＊B

② A/B で商 Q と余り R を求める。

③ A と B の最大公約数を求める。

④分数 A/B を約分する。

⑤(A/B)＊(C/D)の計算　但し A，C は整数

⑥(A/B)＋(C/D)の計算　但し A，C は整数

2　基本的な関数

使用頻度の高い数式も関数としてモジュールにした。

① n の階乗 n!

②順列 ${}_mP_n$

③組合せ ${}_mC_n$

3　桁数の変換

1配列の桁数は使用する計算機によって異なるが、処理速度の関係上なるべく大きい桁数を使用するとよい。単精度、倍精度の桁数は，6，16，32が多いので、1配列の桁数を5の倍数にしておくと桁数変換が容易にできる。

(例)数 A の配列 A(i)の桁数を10桁、B の配列 B(j)の桁数を5桁として、

A = A(1)；A(2)；…；A(N)，

B = B(1)；B(2)；…；B(N)；B(N＋1)；…；B(2N)

と表されているものとする。この時、5桁から10桁への変換は、

A(i)= B(2i-1)＊ E＋B(2i)

となり、10桁から5桁への変換は

B(2j-1)= INT(A(j)/E)

B(2j)= A(j)-INT(A(j)/E)＊ E

となる。

IV　応用

基本演算モジュールを実際に用いていくつかの数式処理プログラム作成したのでその結果を紹介しよう。

1　連立1次方程式

ガウスの消去法を用いて倍精度による近似解と多倍精度による厳密解を計算した。

［例題］

$$\begin{cases} 3046X_1+4567X_2+1304X_3+1011X_4 \\ \quad +16664X_5+86502X_6-71054X_7=-3672 \\ 3896X_1+6455X_2-34674X_3+4561X_4 \\ \quad -78444X_5+10632X_6-23454X_7=-13891 \\ 4107X_1-8073X_2+1205X_3-2671X_4 \\ \quad -19169X_5+4569X_6-5673X_7=92654 \\ 5673X_1-8105X_2+7521X_3+1673X_4 \\ \quad -34292X_5+76310X_6+8345X_7=5634 \\ -8773X_1+45672X_2+7456X_3-46564X_4 \\ \quad -46782X_5+35195X_6-78435X_7=98654 \\ -3213X_1-2183X_2+1057X_3-7463X_4 \\ \quad -73887X_5+64830X_6+8123X_7=-45632 \\ -7254X_1+9124X_2+8645X_3-6015X_4 \\ \quad -92333X_5-4563X_6+82325X_7-78465 \end{cases}$$

近似解

$$\begin{cases} X_1=10.28693959520142 \\ X_2=2.980158314962054 \\ X_3=13.74723888373601 \\ X_4=8.989467633249956 \\ X_5=-3.917673968458871 \\ X_6=-3.267773269383899 \\ X_7=-3.832604639367551 \end{cases}$$

厳密解

$$\begin{cases} X_1=\dfrac{490\ 2546001244\ 9431519216\ 1710932853}{47\ 6579643136\ 2197165122\ 1647250824} \\ X_2=\dfrac{142\ 0282786234\ 0538209705\ 1547878023}{47\ 6579643136\ 2197165122\ 1647250824} \\ X_3=\dfrac{327\ 5827100659\ 6350946612\ 7796801045}{23\ 8289821568\ 1098582561\ 0823625412} \\ X_4=\dfrac{428\ 4197276638\ 8606612309\ 7308691309}{47\ 6579643136\ 2197165122\ 1647250824} \\ X_5=-\dfrac{93\ 3541830906\ 0929491861\ 2812748915}{23\ 8289821568\ 1098582561\ 0823625412} \\ X_6=-\dfrac{77\ 8677109286\ 5281122279\ 7313625615}{23\ 8289821568\ 1098582561\ 0823625412} \\ X_7=-\dfrac{182\ 6541351312\ 0069164695\ 0160684441}{47\ 6579643136\ 2197165122\ 1647250824} \end{cases}$$

2 ベルヌーイ数

k 乗和の公式

$$1^k+2^k+\cdots+n^k$$

等にでてくるベルヌーイ数 B_n を計算した。

B_n の簡単な算出方法には、次の 2 種類ある。

$$\text{① } Bn=\frac{(-1)^{n-1}}{2n+1}\left(\sum_{k=1}^{n-1}(-1)^k\binom{2n+1}{2k}B_k+n-\frac{1}{2}\right)$$

$$\text{② } Bn=(-1)^n\sum_{m=1}^{2n}\sum_{k=1}^{m}\frac{(-1)^k}{m+1}\binom{m}{k}k^{2n}$$

①は n より小さい B_k を使用するので、n =1から連続して B_n を計算するときに用いられる。一方、②は必要なものだけを算出したいとき使用される。どちらの場合にせよ、n が50をこえるとパソコンレベルでは 1 日かかっても計算は終わらない。ここでは、①により B_{70} を書いておく。

［例］

n =70のとき

分子= 328949 0986435898 8039306995 4885188400
6880537476 5735402718 9084548325 2040149898
1532536367 1541455837 4010095577 5033802628
5902471069 6601946049

分母= 679470

3 REDUCE3. 4との比較

数式処理ソフトの REDUCE3. 4と本モジュールとの計算限界に関する比較を行ったので、それを表にして示す。

但し、n 元とは係数 5 桁の n 元連立 1 次方程式のこと。

表 1 計算量比較

計算式	REDUCE3. 4	本モジュール
n !	122 !	8000 !
B_n	B_{47}	B_{70}
n 元	11	11

V おわりに

精度をよくすれば処理時間が多くかかる。今回のプログラムは厳密解を解くことに重点をおいたので処理効率は余りよくない。今後は処理効率を考えたプログラムを作成したいと思う。ただし、本当に速い計算をするためにはワークステーションが必要である。単純に比較してもHP730はパソコンより900倍処理速度が速い。

参考文献

(1) 木田佑司：自作の数学ソフト紹介、数学、1992年7月 P94

(2) A. C. Hearn : REDUCE User's Manual Version 3. 4, 91/7 RAND Publication

(3) D. E. Knuth（島内剛一監訳）: The Art of Computer Programming, 第4分冊, 準数値算法、サイエンス社、1989年