

多倍長ライブラリによる精度評価と改善に関する考察

濱口信行

nobuyuki.hamaguchi.sa@hitachi.com

(株)日立製作所 ソフトウェア事業部

概要

精度評価と改善のため、4倍精度以上の演算を行う多倍長ライブラリを試作し、幾つかの数値計算問題で検討した結果、加減算で発生する誤差要因を定量化した誤差情報、浮動小数点数を表現する配列整数テーブルの使用が非常に有効である事が判明した。

Consideration about precision evaluation and improvement of accuracy
on numeric calculation using Multi-precision library

Nobuyuki Hamaguchi

Software Division, Hitachi, Ltd.

Abstract

We have developed the Multi-precision library that calculates floating point data of quadruple or higher precision in order to improve calculation accuracy. We examined the library on several numeric problems and obtained the result that the use of the array integer table, which represents value of floating point number, and the use of error information on addition and subtraction are very effective on precision evaluation and improvement of accuracy.

1. はじめに

数値計算においては、得られた解が想定される解と明らかに異なっていたり、どの程度の精度があるか不明な場合がよく発生する。しかし、現状では、原因を究明する方法が不明であったり、より高精度の演算の使用による演算時間の増大などの問題がある。本論文では、誤差の解析、精度の改善のための、効率の良い方法を検討した。

なお、浮動小数点数の表現形式はIEEE754形式をとり、 $P \geq 3$ の P 倍精度として符号部 1ビット、指数部 15ビット、仮数部 $32 * p - 16$ ビットを仮定している。

2. 誤差要因

浮動小数点演算で発生する誤差は、

- (1) 近接する2つの数A, B の減算による桁落ち
- (2) 絶対値の差の大きい2つの数A, Bの加算による情報落ち

(3) 乗算、除算の丸め誤差があり、(3)は内積演算の様に、演算結果を更に加減算する場合にも誤差が発生する。

これらの事から、加減算による誤差情報、多くの要素を加算する効率の良い総和計算が必要となる。

3. 加減算

浮動小数点数A, B ($A \neq 0, B \neq 0, A \neq B$) の加減算は、 $A > B > 0$ の $A+B, A-B$ の2ケースに帰着される。

この事から、数値計算で発生する誤差要因を定量化する。(図1)

無効ビット数: $A+B$ において、丸めを含め使用されなかったBのビット数。(情報落ち)

ロストビット数: $A-B$ においてAとC= $A-B$ の指数部の差。(桁落ち)

$A=0, B=0, A=B \neq 0$ の場合は、以下の様に定めている。

無効ビット数: $A=0, B=0, A=B \neq 0$ の場合 0

$A > 0, B=0$ の場合 仮数部のビット数+1

ロストビット数: $A=0, B=0, A > 0, B=0$ の場合 0

$A=B \neq 0$ の場合 仮数部のビット数+2

試作した多倍長ライブラリでは、これらの値を区別するため、無効ビット数には、負符号を付けている。

4. 総和演算

総和計算

T=0

DO I=1, N

T=T+A(I)

END DO

では、A(I)に0があった場合は、加算操作を行わない様にすれば、多倍精度は整数配列IA(1200) とれば演算中にオーバーフロー、アンダーフローは発生しない。この為、浮動小数点数を整数で表現し、演算も整数演算を使用する様にすれば、総和計算は誤差なく計算出来る事になる。

$P_i = A_i * 2^{**}(n) \quad P_i$: 整数 $T = \sum A_i = (\sum P_i) / 2^{**}(n)$
($n=18000$)

5. 数値計算例

数値計算で使用されている幾つかの問題で、誤差情報の適用を検討した。

ここではより一般的で現実的な状況を想定し、個々の数値計算上の技法は使用されていない事を前提としている。

5.1 四則演算

たとえば、4倍精度で、f の値を計算すると、[1]
a=77617.0 b=33096.0

$f = 333.75 * (b^{**}6) + (a^{**}2) * [11 * (a^{**}2) * (b^{**}2) - (b^{**}6) - 121 * (b^{**}4) - 2] + 5.5 * (b^{**}8) + a / (2 * b)$
1.17260394005317863185883490452018 の結果が得られる。

ところが、この計算の理論値は、 $-54767/66192$
 $= -0.827396059946821368141165095497981370$
で異なる結果となっている。4倍精度でのビット消失情報は

bit1...	0
bit2...	-23
[11*(a**2)*(b**2)-(b**6)]-121*(b**4)	
bit3...	-88
[11*(a**2)*(b**2)-(b**6)-121*(b**4)]-2	
bit4...	0
bit5...	114
[333.75*(b**6)+(a**2)*[11*(a**2)*(b**2)-(b**6)-121*(b**4)-2]]+5.5*(b**8)	
bit6...	0

となっていて、bit5 の演算で、[] 内の演算結果と $5.5 * (b^{**}8)$ の加算で0 となる事を示している。また、bit6 の $a / (2 * b)$ を加える演算での値は0であるため、bit5 まで(bit5 も含む)の演算で発生する丸め誤差に原因がある事がわかる。

また、bit2, bit3 での値が負で、-23, -88 より8倍精度だと情報落ちの影響は防げる事を示している。

試作した多倍長ライブラリの8倍精度ルーチンを使用すると、
bit5 121, answer=
 $-0.8273960599468213681411650954979816$

となり、正しい結果となっている。またロストビット数が121 となっているので、4倍精度では正しく計算出来ない事も示している。

5.2 Hilbert 行列の求解

需要予測、回帰分析などで最小二乗近似法を使用する場合、正規方程式は、Hilbert行列Aとなり、 $Ax=b$ の連立一次方程式を解く事が必要になる。ただ、この連立一次方程式は小さな次元数($n=10$ 程度)でも非常に精度が悪くなる事が知られている。[2]

実際、条件数を幾つか計算すると、

n=5	0.943*1.0q+6
n=10	0.353*1.0q+14
n=20	0.628*1.0q+29
n=30	0.118*1.0q+45
n=40	0.225*1.0q+60
n=80	0.324*1.0q+121

となり、得られた結果の精度が悪くなる事を裏付けている。この方程式の求解に誤差情報を適用する。

例として4倍精度で以下の方程式を解いてみると
Hilbert 行列 A :30*30 matrix:Aij=1.0/(i+j-1)
ベクトル b 要素数 30 b(i)=ΣAij (j=1,30)
Ax=b の連立一次方程式を解くと、理論解は
x(i)=1.0 (i=1,30)

結果は、 GOSAMAX= 29.755
ibit = -6 <= b を作成
する場合の無効ビット数の最大値
ibit1= -11 <= 前進消去、
後退代入での無効ビット数の最大値の和

となり、計算結果は、全く信用出来ないものとなっている。

Aij を求める際、除算の丸め誤差が発生するが、Aij の要素から、解がすべて1になる様にbを生成しているの
で、Aij を求める際の除算の丸め誤差は結果の誤差には影響していない。この誤差情報から、Aij を仮数部
112-6-11-1=94 ビットで作成すると、ベクトルbの作成時の
情報落ちの影響が軽減される事が考えられる。Aij の
仮数部を94 ビットで実行すると、
ELAPSE TIME= 2.928000000000000E-003 SEC
GOSAMAX= 9.1958 E-0005となっている。

Aij を4倍精度で求め、b の作成、連立一次方程式の求
解部分を高精度で行うと、プログラム修正に要する負担
は少なく済む。実際この問題で、試作した多倍長ライ
ブラリの8倍精度ルーチンを使用すると
ELAPSE TIME= 9.760000000000000E-003 SEC
GOSAMAX= 2.4428E-0012
となり、実用に充分耐えうる精度となり、演算時間の
増大も、4倍精度の3.5倍以内と負担の少ないものとな
っている。

5.3 特殊関数値の計算

$Li_3(z) = \sum (1/n^{**3}) * z^{**n}$ の様に、特殊関数は、無限
級数の和で表される事が多い。[3]

この場合、必要精度でどこまでの項を加えて良いか
が問題となる。これを、無効ビットを使用して判定する
方法を以下に示した。

4倍精度の範囲で、 $Li_3(1/2)$ の計算例では、

i, ibit=	90	-108
i, ibit=	91	-109
i, ibit=	92	-110
i, ibit=	93	-111
i, ibit=	94	-112

i, ibit= 95 -113

res=0.537213193608043587120848332322414
となり、n=95 まで加えると、それ以降は加えても4倍精
度では値は変化しない事
がわかる。またn=95 までにも、情報落ちが発生してい
る事がわかるので、4倍精度の結果を得るには、8倍精
度ルーチンを使用する必要がある事がわかる。試作した8
倍精度ルーチンでの結果は、以下の様になっている。

結果
i, ibit= 90 -108
i, ibit= 91 -109
i, ibit= 92 -110
i, ibit= 93 -111
i, ibit= 94 -112
i, ibit= 95 -113

res=0.537213193608040200940623225594966
上記2ケースの結果より、4倍精度のまま計算すると10進
14桁までしか精度が保証出来ない事がわかる。

5.4 反復計算

数値解析では、反復計算が多く見られ、この処理では、
値が収束したかどうか判定するのに、相対誤差、絶対
誤差[4]で評価する為、浮動小数点除算がたびたび現
れる。

ここでは、絶対値最小の固有値を算出する場合を例
にとる。[5]

$\theta^2 u / \theta x^2 = -\lambda u$ ($0 < u < 1$) $u(0) = 0, u(1) = 0$ を4
倍精度で計算する。この場合、
理論解 π^{**2}
N等分離散化した時の解 $2^{**N} * 2 * (1 - \cos(\pi/N))$
(計算機理論解) が知られている。

通常収束判定処理は以下の様になる。

```

ERROR=ABS((ENEW-EVAL)/ENEW) <= 除算を使用
EVAL=ENEW
NITER=NITER+1
IF(NITER.EQ.M) GO TO 5
IF(ERROR.GT.EPS) GO TO 2
5 CONTINUE

```

これを、マストビット数で収束判定処理をするには、
CALL Q4ADDSUB(ENEW, EVAL, WORK, 2, ibit)
<= ENEW-EVAL の値をWORK, 誤差情報を ibit にいれ
る。
EVAL=ENEW

精度を改善した総和計算を行うと、

answer=

-0.827396059946821368141165095479816

となり、正しい結果となる。

性能面でも、

Intel Xeon 3.06GHz (2cpu L2キャッシュ (512KB/cpu),

Os: Red Hat Enterprise Linux ES release 3 (taroon)

Kernel: 2.4.21-4 Elsmo on an i686

Intel Fortran V8.0 -02 (注-1)で1cpuで実行

した結果、要素数 $N=10,000,000$ で通常 の4倍精度演算 0.7 秒、精度改善4倍精度0.41 秒

要素数 $N=1,000,000$ で仮数部の長さを変化させた場合、(図3) と、演算時間の増大という問題も解消されている。

(注-1)

IntelおよびIntel Xeonは、アメリカ合衆国およびその他の国におけるインテルコーポレーションまたはその子会社の商標または登録商標です。

Red Hatは、米国およびその他の国でRed Hat, Inc. の登録商標若しくは商標です。

Linuxは、Linus Torvaldsの米国およびその他の国における登録商標あるいは商標です。

6. まとめ

以上の検討により、試作した多倍長ライブラリ、および含まれる誤差情報、配列整数テーブルの使用は、誤差の解析、精度改善を効率的に行うのに有効な事がわかったが、今後は、さらに高精度のルーチンの追加、誤差解析、移行の労力の削減を検討して行く。

7. 参考文献

- [1] A review on Interval Computation -Software and Applications
Chenyi Hu, Shanying Xu, and Xiaoguang Yang
- [2] 杉原正顕、室田一雄 数値計算法の数理
岩波書店 2003
- [3] J. Fujimoto, M. Igarashi, N. Nakazawa, Y. Shimizu and K. Tobimatsu,
Progress of Theoretical Physics, Supplement No. 100 (1990) 1-379
- [4] 伊理正夫 数値計算 朝倉書店 1981
- [5] 菊地文雄、山本昌宏 微分方程式と計算機演習
山海堂 1991
- [6] 森正武 数値解析 共立出版 2002
- [7] R. Piessens E. de Doncker-Kapenga
C. W. Uberhuber D. K. Kahaner
QUADPACK A Subroutine Package for
Automatic Integration
Springer-Verlag Berlin Heidelberg New York

Tokyo 1983

8. 謝辞

精度検証にあたり御指導いただきました 高エネルギー加速器研究機構 藤本、石川、金子先生、に感謝致します。

