Scilab 用 4 倍精度演算環境を利用した混合精度反復法に対する考察

A note on the mixed precision iterative method
using a quadruple precision arithmetic environment on Scilab

[1)] 齊藤 翼　[2)] 石渡 恵美子　[3)] 長谷川 秀彦
1) 東京理科大学大学院理学研究科，2) 東京理科大学理学部,
3) 筑波大学大学院図書館情報メディア研究科

[1)]Tsubasa Saito　[2)]Emiko Ishiwata　[3)]Hidehiko Hasegawa
[1)]Graduate School of Science, Tokyo University of Science,
[2)]Department of Mathematical Information Science, Tokyo University of Science,
[3)]Guraduate School of Library, Information and Media Studies, University of Tsukuba

1　はじめに

コンピュータにおける計算では，IEEE754 によって規定された倍精度浮動小数点数を用いるのが主流である．しかし，浮動小数点演算において，桁落ちや丸め誤差，情報落ちなどの影響は避けられない．このような誤差の影響を検証するためには多倍長演算が必要であるが，特別なハードウェアのない通常の計算機環境で実現するのは難しい．

手軽に高精度演算を利用する方法として，2 つの倍精度浮動小数点数を利用した，Double-Double（DD）演算がある．DD 演算は擬似的に 4 倍精度演算を実行する方法として，Bailey [2] によって提案された．DD 演算を利用した 4 倍精度演算環境はいくつかの研究があり，C や C++のライブラリとして Bailey の QD [2] や，小武守らの Lis [4]，また近年椋木らによって GPU を利用した実装も報告されている [5]．しかし，これらは特定のハードウェアやコンパイラに依存する点や，プログラムコードの書き換えが必要な点で注意が必要である．

そこで [7] において，DD 演算を用いて簡便に 4 倍精度演算を利用できるような環境として ‘QuPAT(Quadruple Precision Arithmetic Toolbox)’ [6] を，フリーの対話型数値計算ソフトウェア Scilab [9] 上に実装した．QuPAT は次の 3 つの特徴：(1) 4 倍精度演算の利用に，プログラムの変更をほとんど必要としない，(2) 倍精度演算と 4 倍精度演算，それらの混合演算を同時に実行できる，(3) ハードウェアや OS には依存しない，を持つ．また [8] で，QuPAT を用いて連立一次方程式に混合精度反復法を適用し，収束性の検証を行った．[3] の前処理に単精度演算を用いる混合精度反復法に対し，我々は反復法のアルゴリズムのある部分にのみ DD 演算を適用している．

本講演ではまず，DD 演算を用いた QuPAT の特徴について述べ，悪条件行列を係数行列に持つ連立一次方程式に対して，QuPAT を用いて一般化共役残差（Generalized Conjugate Residual，GCR）法の収束性を検証する．具体的には，GCR 法において 2 つのスカラー量 α と β に着目し，β に関する部分のみ DD 演算にすることで，すべて DD 演算で計算した場合とほぼ同等の結果を示す事例を報告する．さらに，GCR 法が収束しない場合，α と β の計算のどの部分に原因があるかについても言及する．

2　QuPAT の特徴

文献 [7] において，DD 演算を用いた 4 倍精度演算環境 ‘QuPAT ’ を，オープンソースでフリーの数値計算ソフトウェア Scilab 上に実装した．本節では DD 演算と QuPAT の特徴についての概略を述べる．

DD の数は，2 つの倍精度浮動小数点数で表現する．実数 a を DD の数 A で表現する場合，上位部 Ahi (a を倍精度に丸めた値) と，下位部 Alo ((a − Ahi) を倍精度に丸めた値) の 2 つに分けられる．このとき，DD の数の符号と指数部は上位部 Ahi のみによって決定される．また，DD 演算はすべて倍精度演算の組み合わせで実装される．すなわち，既存の倍精度演算のみの環境でも 4 倍精度演算を扱える．例えば，DD 加算には 20 回，DD 乗算には 24 回の倍精度演算が必要となる．DD 演算の詳細は [7] にある．

DD 演算を利用して，QuPAT は次の 3 つの特徴を持つように実装している．

- 4 倍精度演算の利用に，プログラムの変更をほとんど必要としないこと．
- 倍精度演算と 4 倍精度演算を同時に実行できること．
- ハードウェアや OS には依存しないこと．

Scilab には新しいデータ型を定義でき，演算子をオーバーロードできる機能があり，これを利用して DD 演算を組み込んでいる．

Scilab では倍精度浮動小数点数がデータ型 ‘constant’ によって定められ，スカラー，ベクトル，行列がすべて同様に ‘constant’ として扱われる．そこで Scilab 関数 ‘tlist’ を利用して，DD の数のための新しいデータ型 ‘dd’ を定義した．上述の ‘constant’ の性質から，新しくデータ型 ‘dd’ を定義するだけでスカラー，ベクトル，行列のすべてを DD の数へ自然に拡張できた．加えて ‘dd’ に対する四則演算として DD 演算を用いて，演算子のオーバーロードを適用し，‘constant’，すなわち倍精度数と同じ演算子 (+, −, ∗, /) で四則演算を行えるようにした．

図 1 は，Scilab と QuPAT の関係を表している．QuPAT によって，‘constant’（倍精度数）と ‘dd’（DD の数）は同じ演算子で計算でき，かつ同時に利用できる．‘constant’ と ‘dd’ データ型の変換は，‘d2dd’ や ‘DD.hi’ で行える．通常の Scilab の倍精度演算環境（図 1 の左）は，QuPAT によって DD 演算や，倍精度数と DD の数の混合精度演算環境へ自然に拡張される．すなわち，QuPAT を利用することで従来の Scilab コードからほとんど変更なく DD 演算を利用できる．なお，QuPAT は Scilab の言語機能のみで作成されているため，Scilab が動作する環境であれば，ハードウェアや OS には依存しない．

3 QuPAT を用いた GCR 法の収束性に対する考察

GCR 法は，非対称連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ に対するクリロフ部分空間反復法の 1 つであり，理論的には各反復で残差ノルム $\|\boldsymbol{r}_k\| = \|\boldsymbol{b} - A\boldsymbol{x}_k\|$ が減少し，高々n 回の反復で収束す

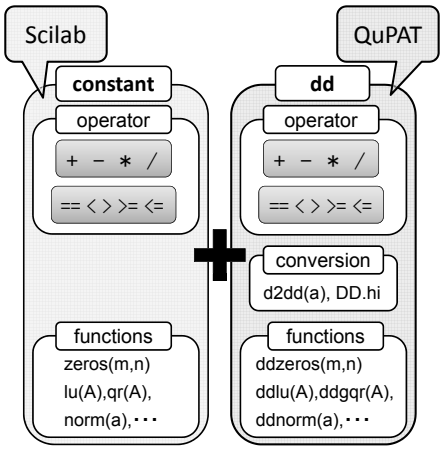

図 1: Scilab と QuPAT の関係

1. Let $\boldsymbol{x}_0$ be an initial guess.
2. set $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\boldsymbol{p}_0 = \boldsymbol{r}_0$, $\boldsymbol{q}_0 = A\boldsymbol{p}_0$, $k = 0$
3. while $\|\boldsymbol{r}_k\|_2 < \varepsilon\|\boldsymbol{r}_0\|_2$ and $k < n$ do
4. $\quad \alpha_k = (\boldsymbol{r}_k, \boldsymbol{q}_k)/(\boldsymbol{q}_k, \boldsymbol{q}_k)$
5. $\quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$
6. $\quad \boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{q}_k$
7. $\quad$ For $i = 0, 1, \cdots, k$ do
8. $\quad\quad \beta_{k,i} = -(A\boldsymbol{r}_{k+1}, \boldsymbol{q}_i)/(\boldsymbol{q}_i, \boldsymbol{q}_i)$
9. $\quad \boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \sum_{i=0}^{k} \beta_{k,i} \boldsymbol{p}_i$
10. $\quad \boldsymbol{q}_{k+1} = A\boldsymbol{r}_{k+1} + \sum_{i=0}^{k} \beta_{k,i} \boldsymbol{q}_i$
11. $\quad k = k + 1$

図 2: GCR 法のアルゴリズム

る．ここで k は反復回数，n は係数行列 A の次元数である．しかし，浮動小数点演算を用いると収束しない場合がある．

本節では QuPAT を用いて GCR 法の収束性について考察する．数値実験に用いた計算機は CPU : AMD Turion(tm) X2 Dual-Core 2.00GHz，Memory 1.87GB であり，Scilab のバージョンは 5.1.1 である．初期ベクトルは $\boldsymbol{x}_0 = (0, 0, \cdots, 0)^T$，右辺ベクトル $\boldsymbol{b}$ は解 $\boldsymbol{x}^*$ の値がすべて 1 となるように定める．最大反復回数は次元数 n とする．GCR 法のアルゴリズムを図 2 に示す.

3.1 倍精度演算と DD 演算の計算結果の比較

まず，倍精度演算のみを用いた場合と DD 演算のみを用いた場合の計算結果を比較する．以後，‘double’ は倍精度演算のみを用いた計算，‘DD’ は DD 演算のみを用いた計算を表すものとする．停止条件は ‘double’ : $\|\boldsymbol{r}_k\|_2 \leq 10^{-12}\|\boldsymbol{r}_0\|_2$，‘DD’ : $\|\boldsymbol{r}_k\|_2 \leq 10^{-14}\|\boldsymbol{r}_0\|_2$ とする．係数行列には MatrixMarket から arc130，pores_3 を用いる．arc130 の次元数は 130，条件数は 6.05×10^{10} であり，pores_3 の次元数は 532，条件数は 5.61×10^{5} である．表 1 に ‘double’ と ‘DD’ の計算結果を示す．反復回数を N とする．表中の相対残差は $\|\boldsymbol{b} - A\boldsymbol{x}_N\|_2/\|\boldsymbol{b} - A\boldsymbol{x}_0\|_2$ を表し，相対誤差は $\|\boldsymbol{x}_N - \boldsymbol{x}^*\|_\infty/\|\boldsymbol{x}^*\|_\infty$ を表すものとする．ここでは arc130 を取り上げ，結果を確認する．‘double’ の場合，相対残差が 1.0×10^{-10} 程度で停滞してしまい，相対誤差が 1.81×10^{0} となり 1 桁も正しい解が得られない．一方で ‘DD’ の場合，15 回の反復で相対残差は 7.15×10^{-16}，相対誤差が 2.40×10^{-6} となり，理論通りに残差が減少した（図 3）.

表 1: ‘double’ と ‘DD’ の計算結果

	‘double’			‘DD’		
	反復回数 N	相対残差	相対誤差	反復回数 N	相対残差	相対誤差
arc130	†	7.31e-11	1.82e+00	15	7.15e-16	2.40e-06
pores_3	†	3.16e-06	3.06e-02	474	9.48e-15	3.81e-12

† : No convergence.

3.2 α と β に着目した混合精度 GCR 法

‘DD’ の場合，‘double’ に比べて残差と誤差の精度が改善されることを前節で確認した．しかし，すべての計算を DD 演算で行うと，倍精度演算に比べて多くの計算時間とメモリ量が必要となる．そこで，アルゴリズムの一部分に DD 演算を適用する混合精度反復法によって，収束性が向上するか調べる．GCR 法において 2 つのスカラー量 α と β に着目し，次の 2 つの方針を与えて比較する．

(i) α_k，$\boldsymbol{x}_{k+1}$，$\boldsymbol{r}_{k+1}$ に DD 演算を用い（図 2 の 4〜6 行目），α_k と $\boldsymbol{r}_{k+1}$ を DD の数とする．

(ii) $\beta_{k,i}$，$\boldsymbol{p}_{k+1}$，$\boldsymbol{q}_{k+1}$ に DD 演算を用い（図 2 の 7〜10 行目），$\beta_{k,i}$ と $\boldsymbol{q}_{k+1}$ を DD の数とする．

表 2: 方針 (i) と (ii) の計算結果

	方針 (i)			方針 (ii)		
	反復回数 N	相対残差	相対誤差	反復回数 N	相対残差	相対誤差
arc130	†	8.82e-11	2.29e+00	15	3.17e-15	2.40e-06
pores_3	†	3.17e-06	3.07e-02	474	1.47e-14	4.05e-12

† : No convergence.

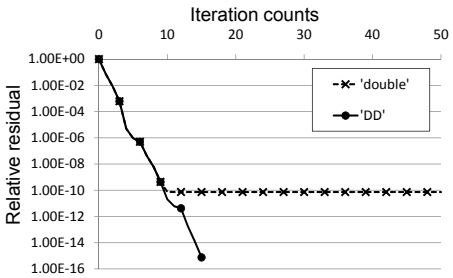

図 3: arc130 の収束履歴

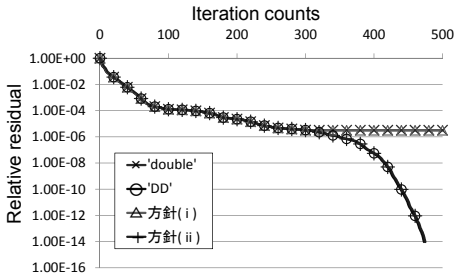

図 4: pores_3 の収束履歴

表 2 に方針 (i) と (ii) の計算結果を示す．停止条件はどちらの方針も $\|\boldsymbol{r}_k\|_2 \leq 10^{-14}\,\|\boldsymbol{r}_0\|_2$ とする．方針 (i) の場合，どちらのテスト行列に対しても相対残差の値は改善しなかった．これに対して方針 (ii) の場合は，相対残差の値が改善し，‘DD’ の場合とほぼ同等の収束性を示した(図 4)．

このように，方針 (ii) を用いること，すなわち $\beta_{k,i}$，$\boldsymbol{p}_{k+1}$，$\boldsymbol{q}_{k+1}$ のみを DD 演算で計算することで GCR 法の収束性が改善し，すべての計算に DD 演算を用いた場合とほぼ同等の結果を示す事例があるとわかった．‘DD’ の場合と方針 (ii) の場合で反復回数が等しい arc130 に対して，収束までの計算時間は ‘DD’ で 8.27 秒，方針 (ii) で 1.92 秒となり，1 / 4 に短縮された．また，メモリ使用量についても ‘DD’ の場合は ‘double’ に対して 2 倍必要となるが，方針 (ii) の場合は $\boldsymbol{q}_{k+1}$ のみ DD の数で保持するため，‘double’ に対して約 1.5 倍で済む．よって，方針 (ii) はすべての計算を DD 演算で行うよりも，計算時間の面からもメモリ使用量の面からも有効であると考えられる．

一方，[1] において，α_k が 0 に近い値になるとき，残差の停滞と関連することが報告されている．混合精度演算を用いることで，$\alpha_k \approx 0$ となる原因が数値計算の結果から明らかになっており，講演時には α_k や $\beta_{k,i}$ の計算のどの部分が原因で上記の事例が起きたり，残差の停滞を引き起こすかについて具体的に述べる．

参考文献

[1] K. Aihara, E. Ishiwata and K. Abe, A strategy of reducing the inner iteration counts for the variable preconditioned GCR(*m*) method, JSIAM Letters, Vol. 2, 77-80 (2010).

[2] D. H. Bailey, QD (C++ / Fortran-90 double-double and quad-double package), Available at http://crd.lbl.gov/˜dhbailey/mpdist/

[3] J. D. Hogg, J. A. Scott, A fast robust mixed precision solver for the solution of sparse symmetric linear systems, Technical Report, RAL-TR-2008-023 (2008).

[4] 小武守恒, 藤井昭宏, 長谷川秀彦, 西田晃, 高速な 4 倍精度演算を用いたクリロフ部分空間法の安定化, 計算工学講演会論文集 Vol. 12, 631-634 (2007).

[5] 椋木大地, 高橋大介, GPU による 4 倍精度 BLAS の実装と評価，計算工学講演会論文集，Vol. 15, No. 2, 891-894 (2010).

[6] QuPAT, http://www.mi.kagu.tus.ac.jp/qupat.html

[7] T. Saito, E. Ishiwata and H. Hasegawa, Development of Quadruple Precision Arithmetic Toolbox QuPAT on Scilab, ICCSA2010, Proceedings Part II, LNCS 6017, 60-70 (2010).

[8] T. Saito, E. Ishiwata and H. Hasegawa, Analysis of the GCR method with mixed precision arithmetic using QuPAT, submitted to Journal of Computational Science.

[9] Scilab, http://www.scilab.org/