MATLAB 版
MuPAT ユーザーガイド

八木武尊

2019 年 4 月 30 日

目次

1 はじめに

MuPAT は Scilab[1]，MATLAB 上 [2] に実装した擬似 4 倍精度演算 (`double-double`)[3]，擬似 8 倍精度演算 (`quad-double`)[4] が扱える高精度演算環境です．現在は MacOS のみ提供しています．MuPAT では MATLAB コマンドウィンドウを用いて対話的に，また，倍精度 (`double`) のコードを変えることなく，簡単に上記の高精度演算を扱うことができます．さらに，MuPAT では頻出するベクトル演算と行列演算に FMA, AVX2, OpenMP を用いた並列処理を適用しており，高速な処理ができます．並列処理の詳細についてはユーザーガイドの並列処理の項目を参照してください．

1.1 システム要件

MuPAT は以下の環境があればすべての機能が利用可能です．

Mac OS
MATLAB R2017a 以降
CPU は intel では Haswell 以降，AMD は調査中

2 MuPAT 起動方法

2.1 インストール

MuPAT をホームページ [https://www.ed.tus.ac.jp/1419521/] もしくは，[https://github.com/HotakaYagi/] MuPAT から zip ファイルをダウンロードし，解凍した後，MATLAB ディレクトリ下など，作業しやすいディレクトリへ保存してください．

2.2 起動

1. MATLAB を起動
2. MATLAB 上でカレントディレクトリを MuPAT ディレクトリへ移動
3. MATLAB コマンドウィンドウで以下のコマンドを入力する
   `>> startMuPAT(n1, n2, n3)`

その後 `a = DD(1)` と MATLAB コマンドウィンドウで入力し，次の結果が出力されれば MuPAT は正常です．

```
>> a = DD(1)
a =
hi:  1
lo:  0
```

MuPAT は入力引数としてスレッド数, AVX2 の on/off, FMA の on/off を選べます．(引数なしはスレッド数 = 1 で FMA, AVX2 はオフです) FMA，AVX2，スレッド数の説明についてはユーザーガイドの並列処理の項目を参照してください．

例)
`>> startMuPAT`: スレッド数=1, AVX2 無効, FMA 無効．(逐次計算)
`>> startMuPAT(2,1,1)`: 2 スレッドで AVX2 有効，FMA 有効．

表 2.1　startMuPAT への入力引数

	スレッド数 (n1)	AVX2 (n2)	FMA (n3)
高速化 ON	$\{n \mid n \in \mathbb{N}, n > 1\}$	1	1
高速化 OFF	1 (逐次) or 引数なし	0 or 引数なし	0 or 引数なし

2.3　注意点

問題の要求メモリサイズに気をつけてください．MATLAB の mex 機能は要求メモリサイズが大きいと MATLAB が強制終了してしまうことがあります．使用しているスレッド数によっても変わります．
さらに，並列化を使用した際には計算順序が逐次の時と変わる場合があり，丸め誤差の影響で計算結果が逐次のときと比べて完全に一致はしない場合があります．

3　使用方法

MuPAT では，擬似 4 倍精度演算と，擬似 8 倍精度数演算を行うことができます．これらの演算は倍精度演算と同様のコマンドでスカラーやベクトル，行列演算などを扱うことができます．MuPAT 特有の操作は以下の 3 点です．

- DD 型、QD 型の宣言または変換 (3.1 節から 3.2 節).
- DD 型、QD 型の入出力 (3.3 節).
- DD 型、QD 型の関数呼び出し (3.4 節).

3.1　変数の定義

MATLAB 上での数値はすべて倍精度 (double) で扱われています．MuPAT 上では，擬似 4 倍精度演算を扱える DD 型という新しいデータ型を，擬似 8 倍精度演算を扱える QD 型という新しいデータ型を定義しています．DD 型は擬似 4 倍精度計算のために hi(上位パート) と lo(下位パート) の 2 つの double 型変数を，QD 型は擬似 8 倍精度計算のために hh(最上位パート) から ll(最下位パート) までそれぞれ hh, hl, lh, ll の 4 つの double 型変数を持ちます．DD 変数 A の上位パート hi や下位パート lo の値はそれぞれ A.hi, A.lo と記述することで参照でき，QD 変数 B も最上位から最下位までそれぞれ B.hh, B.hl, B.lh, B.ll と記述すれば値を参照できます．

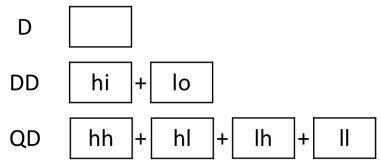

図 3.1　DD 型の数と QD 型の数のイメージ

DD 型を定義するためには以下の DD 関数が必要です．DD 関数は引数を 2 個まで定義できます．入力に関してスカラー・行列は区別しません．

```
>> DD(a, b)
```

- DD() のように引数が 0 個なら上位パート hi と下位パート lo には値 0 が割り当てられます.
- DD(a) のように引数が 1 個の場合は a が上位パート hi へ割り当てられ，下位パート lo には値 0 が割り当てられます.
- 引数が 2 個の場合，上位パート hi と下位パート lo に入力した値 a, b が割り当てられます.

QD 型を定義するためには以下の QD 関数が必要です．QD 関数は引数を 4 個まで定義できます．入力に関してスカラー，行列は区別しません.

```
>> QD(a, b, c, d)
```

- QD() のように引数が 0 個なら最上位パート hh から最下位パート ll までそれぞれ値 0 が割り当てられます.
- QD(a) のように引数が 1 個の場合は最上位パート hh 以外，つまり b 以降の値 (hl, lh, ll) には値 0 が割り当てられます.
- QD(a, b) のように引数が 2 個の場合，上 2 つの位 (hh, hl) に a, b が，下 2 つの位 (lh, ll) に値 0 が割り当てられます.
- QD(a, b, c) のように引数が 3 個なら上 3 つの位 (hh, hl, lh) に a, b, c が，最下位パート ll には値 0 が割り当てられます.
- 引数が 4 つの場合は，最上位パート hh から最下位パート ll に入力した値 a, b, c, d が割り当てられます.

例)

```
>> a = DD(1, 0)
a =
hi:  1
lo:  0
>> b = QD(3.14, 0, 0, 0)
b =
hh:  3.1400
hl:  0
lh:  0
ll:  0
>> A = DD([1, 2, 3; 4, 5, 6])
A =
hi:  [2x3 double]
lo:  [2x3 double]
>> A.hi
ans =
1 2 3
4 5 6
>> A.lo
ans =
0 0 0
0 0 0
```

3.2 型の変換

double 型変数 a，DD 型変数 b，QD 型変数 c のデータの型を変換したいときは以下のようにしてください．

表 3.1 型の変換方法

from\to	double	DD	QD
double		DD(a)	QD(a)
DD	double(b)		QD(b)
QD	double(c)	DD(c)	

例)
DD 型変数 b を double 型にしたい場合は
>> double(b)
double 型変数 a を QD 型にしたい場合は
>> QD(a)

3.3 入出力

MuPAT には次の入出力関数があります．この関数はスカラー変数のみ使用できます．

- ddinput(a)
  文字で表される数値列を DD 型の数値へ変換します．
- qdinput(a)
  文字で表される数値列を QD 型の数値へ変換します．
- ddprint(a)
  十進 31 桁で文字列として DD 型の変数を表示します．
- qdprint(a)
  十進 63 桁で文字列として QD 型の変数を表示します．

3.4 その他の DD 型，QD 型関数

擬似乱数の作成方法は，例えば DD 型では上位パート，下位パートそれぞれで MATLAB 関数によって乱数を作り，正規化して 1 つの値としています．QD 型も同様に最上位から最下位までそれぞれ MATLAB 関数によって乱数を作り，正規化して 1 つの値としています．

- ddrand(m, n)
  m×n 型の DD 型擬似乱数行列を返します．
- qdrand(m, n)
  m×n 型の QD 型擬似乱数行列を返します．
- ddeye(m, n)
  m×n 型の DD 型単位行列を返します．
- qdeye(m, n)
  m×n 型の QD 型単位行列を返します．

3.5 算術演算

3.5.1 四則演算子

DD 型，QD 型，と混合精度演算では四則演算子 (+, -, *, /) を使用することができます．除算はスカラーのみ対応しています．型の異なる高精度演算をするときは，計算結果は精度が高い方の型で保存されます．

例 1)

```
>> d = 1 + 1.0E-20
d =
1
>> e = DD(1, 1.0E-20)
e =
hi:  1
lo:  1.0000e-20
>> e - d
ans =
hi:  1.0000e-20
lo:  0
```

例 2)

double 型 , DD 型，QD 型の混合精度演算を使用して 1 ÷ 3 = 0.333…を計算します．

```
>> format long
>> disp(1/3);
0.333333333333333
>> ddprint(DD(1)/3);
' 3.33333333333333333333333333333E-1'
>> qdprint(QD(1)/3);
'3.33333333333333333333333333333333333333333333333333333333333333E-1'
```

※コマンドウインドウが小さいと，出力の一部が省略される場合があります．

3.5.2 ドット演算子

DD 型，QD 型でも MATLAB と同じようにドット演算 (.* , ./) を使用することができます．

例 3)

```
>> A = DD([1.1, 1.2, 1.3]);
>> B = DD([1.1, 1.2, 1.3]);
>> C = A .* B
hi:  [1.2100 1.4400 1.6900]
lo:  [8.8818e-18 -5.3291e-17 -5.3291e-17]
```

例 4)

```
>> A = DD([1.1, 1.2, 1.3]);
>> B = DD([1.1, 1.2, 1.3]);
```

```
>> C = A ./ B
ans =
hi:  [1 1 1]
lo:  [0 0 0]
```

3.6 関係演算子

DD 型, QD 型でも MATLAB と同じように論理演算子 $(=, \sim=, <, >, \leq, \geq)$ を使用することができます.

例 5)

```
>> a = 1/7;
>> b = 1/DD(7);
>> c = 1/QD(7);
>> a == b
ans =
logical
0
>> a < b
ans =
logical
1
>> a ~= c
ans =
logical
1
>> a == b.hi
ans =
logical
1
```

3.7 論理演算子

DD 型, QD 型でも MATLAB と同じように論理演算子 $(\&, |, \sim)$ を使用することができます.

例 6) DD 型での論理演算子

```
>> A = DD([1,0,3; 4,5,0]);
>> B = DD([0,0,3;4,0,6]);
>> A & B
ans =
2x3 logical array
0 0 1
1 0 0
>> A | B
ans =
2x3 logical array
```

```
1 0 1
1 1 1
>> ~A
ans =
2x3 logical array
0 1 0
0 0 1
```

3.8 配列における要素の代入や抽出

- `A(i, j) = b`
  配列 A の (i, j) 要素に変数 b の値を代入します．左辺の精度が右辺よりも高くないと代入は反映されません．

例)

```
>> A = DD([1,2,3; 4,5,6]);
>> A(2, 3)
ans =
hi:  6
lo:  0
>> A(2, 2) = 10;
>> A.hi
ans =
1  2 3
4 10 6
>> A.lo
ans =
0 0 0
0 0 0
```

>> A(2, 2) = QD(1); これは A が DD 型配列なため QD 型要素を代入できずにエラー

3.9 その他の MATLAB 関数

DD 型, QD 型でも MATLAB で定義されたいくつかの関数を使用することができます．

3.9.1 サポートしている関数

以下の操作は MATLAB と同様に扱うことができます．

- `A'`
  A の転置行列を返します．
- `[A, B]`
  A, B を水平方向に連結します．
- `[A; B]`
  A, B を垂直方向に連結します．
- `[m, n] = size(a)`
  行列 a の行の長さ m と列の長さ n を返します．

- abs(a)
  行列 a の絶対値を返します.
- sqrt(a)
  行列 a の平方根を返します.
- dot(x, y)
  ベクトル x と y の内積を返します.
- tmv(a, b)
  行列 a' とベクトル b の転置行列積を返します.
- a^n
  スカラー a を n 乗した値を返します.
- norm(a,n)
  a の n ノルムを返します. n は 1, 2, inf, fro が想定されます.

3.9.2 サポートしていない関数

3.9.1 節で述べた関数以外の関数，ならびに数値解法や行列の分解などの関数はサポートしていません．もしサポートしていない MATLAB 関数を呼び出すと以下のようになります.

例)

```
>> a=ddrand(5);
>> qr(a)
Undefined function 'qr' for input arguments of type 'DD'.
```

3.10 Scilab 版 MuPAT との違い

Scilab 版では DD 型，QD 型を宣言する際，小文字で dd，qd と記述していましたが，MATLAB 版では大文字で関数を呼び出す必要があります．型を変換する際に Scilab 版では，例えば，倍精度から DD 型にしたいときには d2dd() 関数，倍精度から QD 型にしたいときには d2qd() 関数を呼び出す必要がありましたが，MATLAB 版では変更したい変数を変更したい型へ代入すれば良いです．詳しくはユーザーガイドの型の変換 (3.2 節) を参照してください．また，DD 型，QD 型の上位パートなどの倍精度要素を取り出す際，Scilab 版では getHi() 関数を呼び出す必要がありましたが，MATLAB 版では DD 型変数 a では a.hi，QD 型変数 b では b.hh などと記述すれば倍精度要素を適宜取り出せます.

また，Scilab 版でサポートしていた疎行列形式，数値解法や，行列の分解などの関数 (詳しくは 3.9.1 節と 3.9.2 節を参照してください) は MATLAB 版ではサポートしていません.

4 並列処理

FMA, AVX2, OpenMP を有効または無効にする際には，以下のようにコマンドを実行します.

```
>> startMuPAT(n1, n2, n3)
```

として引数 n1 (OpenMP のスレッド数 (off はスレッド数=1 に相当))，n2 (AVX2 の on/off)，n3 (FMA の on/off) で指定ください．指定しなかった場合は指定されなかった機能が off となります.

n1 は非負数の入力が想定され，n2，n3 は 0 (off) か 1 (on) の入力をしてください.

MuPAT 起動時以外にも，以下のようにして並列処理の on/off を切り替えられます.

```
>> fmaon
>> fmaoff
>> avxon
```

```
>> avxoff
>> omp(n1)
```
現在の高速化の状況を確認するためには statusMuPAT コマンドを用います．

例)
```
>> startMuPAT
>> statusMuPAT
Cureently MuPAT is running at the number of threads:  1thread (serial), AVX is off,
and FMA is off
>> avxon
>> omp(4)
Cureently MuPAT is running at the number of threads:  4, AVX is on, and FMA is off
```

4.1 FMA

FMA(Fused Multiply Add) 演算 [5, 6] は，x = a × b + c 形式の積和演算を 1 演算で実行します．これにより，積と和の組で表される演算の回数を半減でき，最大で 2 倍の性能向上が見込めます．積差についても使えます．FMA を有効にすると FMA 向けの計算アルゴリズム (troprod_fma[4]) が使われます．そのため，FMA を有効にした場合では積演算における内部のアルゴリズムが一部異なります．

4.2 AVX2

Intel AVX2(Advanced Vector Extensions)[5, 6] は，4 つの倍精度浮動小数点数演算を 1 命令で実行できます．最大 4 倍の性能向上が見込めます．

例)
```
>> startMuPAT(1, 0, 0)
>> a = qdrand(500);
>> b = qdrand(500);
>> tic; a * b; toc;
Elapsed time is 8.374103 seconds.
>> avxon
>> tic; a * b; toc;
Elapsed time is 2.227658 seconds.
```

4.3 OpenMP

OpenMP[7] によって並列プログラミングができます．OpenMP ではコア数の分だけ高速化が可能です．ここで，スレッド数とは何並列で処理を実行するかを表しており，最大スレッド数は CPU によって異ります．スレッド数=高速化率とは限りませんが，1 から最大スレッド数の中から適当に選んで実行してください．通常スレッド数=コア数とすると良いでしょう．

例)
```
>> startMuPAT(1, 0, 0)
>> a = ddrand(500);
```

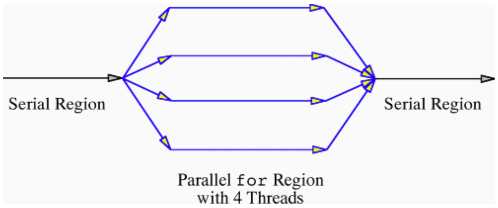

図 4.1　OpenMP によるマルチスレッド処理のイメージ

```
>> b = ddrand(500);
>> tic; a * b; toc;
Elapsed time is 0.814772 seconds.
>> omp(2)
set the number of threads = 2
>> tic; a * b; toc;
Elapsed time is 0.449846 seconds.
```

4.4　高速化されている処理

MuPAT では上記の並列化によって，ある程度大きな処理であれば，例えば 4 コアマシンでは最大で 16 倍から 17 倍程度早く計算ができます．並列化による高速処理が可能な 48 種類の演算を以下の表に記します．表中で，例えば D-DD という記号は倍精度数 x と DD 数 y で何らかの演算 x○y を行うことを意味しており，DD-D は DD 数 y と倍精度数 x で何らかの演算 y○x を行うことを意味しています．また，表での関数呼び出し方法とは，演算 y○x に必要な型 → 演算 y○x を行うためのコマンドを意味しています． なお，現在疎行列形式はサポートしていません．

表 4.1　高速化をサポートしている関数と呼び出し方法

	D-DD	D-QD	DD-D	DD-DD	DD-QD	QD-D	QD-DD	QD-QD
ベクトル (行列) 和 (差)	x, y: vector or matrix → `x + y` or `x - y`							
スカラー倍	a: scaler, x: vector or matrix → `a * x` or `x * a`							
内積	x, y: vector → `dot(x, y)` or `x' * y`							
密行列ベクトル積	A: matrix, b: vector → `A * b`							
密転置行列ベクトル積	A: matrix, b: vector → `tmv(A, b)`							
密行列積	A, B: matrix → `A * B`							

参考文献

[1] S. Kikkawa, T. Saito, E. Ishiwata and H. Hasegawa, Development and acceleration of multiple precision arithmetic toolbox MuPAT for Scilab, JSIAM Letters, Vol.5, pp.9-12 (2013).

[2] H. Yagi, E. Ishiwata, and H. Hasegawa, Acceleration of interactive multi-precision arithmetic toolbox MuPAT using parallel processing, Forum on Information Technology, Vol.1, pp.43-48 (2018).

[3] D. H. Bailey, High-Precision Floating-point arithmetic in scientific computation, Computing in Science and Engineering, Vol.7(3), pp.54-61

[4] Y. Hida, X. S. Li and D. H. Baily, Quad-double arithmetic: algorithms, implementation and application, Technical Report LBNL-46996, Lawrence Berkeley National Laboratory, Berkeley (2000).

[5] Intel: Intrinsics Guide, available from https://software.intel.com/sites/landingpage/IntrinsicsGuide/

[6] Intel: 64 and IA-32 Architectures Optimization Reference Manual, https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-optimization-manual

[7] OpenMP : http://www.openmp.org/

[8] Matlab Documentation: https://jp.mathworks.com/help/matlab/ref/mex.html

Appendix

A: フォルダ構造

MuPAT のフォルダ構成を以下に記します.

- MuPAT
  - README.md
  - useguideJap.pdf
  - useguideEng.pdf
  - @DD
    - DD.m
  - @QD
    - QD.m
  - @Calc
    - Calc.m
  - src
    - ddeye.m
    - ddrand.m
    - qdeye.m
    - qdrand.m
    - avxoff.m
    - avxon.m
    - fmaoff.m
    - fmaon.m
    - omp.m
    - statusMuPAT.m
    - mupat.c（論文提出後）
    - mupat.h
    - .....(set of C program)
    - .....(set of MEX files)