MATLAB 版
MuPAT ユーザーガイド

八木武尊

2019 年 4 月 9 日

目次

1　はじめに

Matlab 上に実装した擬似 4 倍精度数 (double-double)[1]，擬似 8 倍精度数 (quad-double)[2] が扱える高精度演算環境 MuPAT[3] は OS 毎に用意されています (今は Mac のみ)．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) のときのコードをほぼ変えることなく，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へ並列処理を適用しています．並列化手法としては，FMA, AVX2, OpenMP の 3 種類を利用しており，高速な処理が可能となっています．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．

2　インストール

MuPAT toolbox をホームページからダウンロード，もしくは GitHub からダウンロードした後，MuPAT を 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．

2.1　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．

MATLAB R2017a 以降

CPU は intel では Haswell 以降，AMD は調査中

2.2　MuPAT 起動方法

1. MATLAB を起動
2. MATLAB 上で MuPAT ディレクトリへ移動
3. 以下のコマンドを MATLAB コマンドウィンドウに入力する

>>startMuPAT(n1, n2, n3)

※ 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 はオフに設定されます)

例)

startMuPAT(2,1,1): 2 スレッドで AVX2 有効，FMA 有効で計算．

startMuPAT(4,0,1): 4 スレッドで AVX2 無効，FMA 有効で計算．

startMuPAT(1,0,0): 1 スレッドで AVX2 無効，FMA 無効で計算．(逐次に相当)

上記の手順を終えた後 a = DD(1) と MATLAB コマンドウィンドウに入力し，次の結果が出力されていれば MuPAT を使える状態にあります．

>>a = DD(1)

a =

hi: 1

lo: 0

3　使用方法

MuPAT では，倍精度と同じ演算子を利用し，擬似 4 倍精度演算を意味する Double-Double(DD) 演算と，擬似 8 倍精度演算を意味する Quad-Double(QD) 演算を行うことができます．これは MATLAB で新しい型

を定義し，演算子のオーバーロードを行ったためです．DD は hi(上位パート) と lo(下位パート) の 2 つの double 型変数を持ち，QD は最上位パートから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．

3.1 変数の定義

MATLAB 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 倍精度演算の数を得るために DD 型という新しいデータ型を，擬似 8 倍精度演算の数を得るために QD 型という新しいデータ型を定義しています．これらの型では double 型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．次の関数で DD 型，QD 型の変数を宣言します．

>> DD(a, b)
DD 型には 0 個から 2 個の引数が必要です．DD() のように引数が 0 個なら上位パートと下位パートには値 0 が割り当てられ，DD(a) のように 1 個の引数しか入力しない場合は a が上位パートへ割り当てられ，下位パートに自動的に値 0 が割り当てられます．2 つの引数が入力された場合，a, b それぞれに入力した値が割り当てられます．入力 a，b の型はスカラーでもベクトルでも行列でも，a と b の型が一致している限り関係ありません．

>> QD(a, b, c, d)
QD 型には 0 個から 4 個の引数が必要です．QD() のように引数が 0 個なら最上位パートから最下位パートまで値 0 が割り当てられ，QD(a) のようにもし 1 個の引数しか入力しない場合は最上位パート以外，つまり b 以降の値に自動的に値 0 が割り当てられます．QD(a, b) のように 2 個の引数が入力された場合，下 2 つの位に値 0 が，QD(a, b, c) のように引数が 3 個なら最下位パートに値 0 が割り当てられます．4 つの引数が入力された場合は，最上位パートから最下位パートまで a, b, c, d それぞれに入力した値が割り当てられます．
また，DD 型変数 A を QD 型へ QD(A) と代入したら，QD の上 2 つの位に値 DD 型変数 A の上位パートと下位パートが割り当てられ, 下 2 つの位以降の値は自動的に 0 になります．QD 型においても入力 a，b，c，d の型はスカラーでもベクトルでも行列でも，a，b，c，d の型が一致している限り関係ありません．

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 とすれば値を参照できます．
例)
>> a = DD(1, 0)
a =
hi: 1
lo: 0
>> b = QD(1, 0, 0, 0)
b =
hh: 1
hl: 0
lh: 0
ll: 0
>> A = [1, 2, 3; 4, 5, 6];
>> AA = DD(A)
AA =

```
hi: [2x3 double]
lo: [2x3 double]
>> AA.hi
ans =
1 2 3
4 5 6
>> AA.lo
ans =
0 0 0
0 0 0
```

3.2 入出力

MuPAT には次の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．

・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．

3.3 型の変換

MuPAT では次のような関数でデータの型を変換できます．

・double(a)

DD，QD 型の変数 a を double 型に変換します．

・DD(a)

Double 型，QD 型の変数 a を DD 型に変換します．

・QD(a)

Double 型，DD 型の変数 a を QD 型に変換します．

3.4 要素の書き込みや抽出

・A(i, j) = b

DD 型の変数 A の (i, j) 要素に DD 型の変数 b を書き込みます．

・A(i, j)

DD 型の変数 A の (i, j) 要素を抽出します．

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

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

3.5 算術演算子

3.5.1 四則演算子

DD 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．

例 1)
Double 型と DD 型を使用して $1 + 10^{-20}$ を計算します．
>> 1 + 1.0E-20
ans =
1
>> e = DD(1);
>> f = DD(1.0E-20);
>> e + f
ans =
hi: 1
lo: 1.0000e-20

例 2)
Double 型 , DD 型を使用して $1 \div 3 = 0.333\cdots$を計算します．
>> format long
>> f = 1/3
f =
0.333333333333333
>> g = DD(1) / DD(3)
g =
hi: 0.333333333333333
lo: 1.850371707708594e-17
>> ddprint(g)
ans =
' 3.33333333333333333333333333333333E-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　その他の算術演算子

Double 型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)

2x2 の DD 型行列を使用して要素ごとの乗算を行います.

```
>> A = [1,2,3; 4,5,6];
>> B = [2,2,2; 5,5,5];
>> AA = DD(A);
>> BB = DD(B);
>> CC = AA .* BB;
>> CC.hi
ans =
2   4   6
20 25 30
>> CC.lo
ans =
0 0 0
0 0 0
```

例 4)

2x2 の DD 型行列を使用して要素ごとの除算を行います.

```
>> A = [1,2,3; 4,5,6];
>> B = [3,3,3; 3,3,3];
>> AA = DD(A);
>> BB = DD(B);
>> CC = AA ./ BB;
>> CC.hi
ans =
0.333333333333333 0.666666666666667 1.000000000000000
1.333333333333333 1.666666666666667 2.000000000000000
>> CC.lo
ans =
1.0e-16 *
0.185037170770859 0.370074341541719 0
0.740148683083438 -0.740148683083438 0
```

3.5.3　混合型による演算

Double 型と DD 型の可算や double 型と QD 型，DD 型と QD 型の加算もまた，”+” 演算子を使うことによって計算することができます．これは型の混ざった演算が可能なことを意味します．精度の混ざった高精度演算をするとき，計算結果は自動的に精度が高い方の型に変換され結果が出力されます.

3.6 関係演算子

Double 型のように，DD, QD 型でも論理演算子 $(=, \sim=, <, >, \leq, \geq)$ を使用することができます．

例 5) double 型で格納された 1/7 と DD 型，QD 型で格納された 1/7 の関係

```
>> format long
>> a = 1/7;
>> b = 1/DD(7);
>> c = 1/QD(7);
>> disp(a)
hi: 0.1429
lo: 7.9302e-18
>> ddprint( b )
ans =
' 1.428571428571428571428571428571E-1'
>> disp(c)
hh: 0.1429
hl: 7.9302e-18
lh: 4.4021e-34
ll: 2.4437e-50
>> qdprint(c)
ans =
' 1.428571428571428571428571428571428571428571428571428571428571E-1'
>> a == b
ans =
0
>> a < b
ans =
1
>> a ~= c
ans =
1
>> a == b.hi
ans =
1
```

3.7 論理演算子

Double 型のように，DD, QD 型でも論理演算子 $(\&, |, \sim)$ を使用することができます．

例 6) DD での論理演算子

```
>>A = [1,0,3; 4,5,0];
>> B = [0,0,3;4,0,6];
>> AA=DD(A);
>> BB=DD(B);
>> AA & BB
```

ans =
0 0 1
1 0 0
>> AA | BB
ans =
1 0 1
1 1 1
>> ∼AA
ans =
0 1 0
0 0 1

3.8 その他演算子

・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.
例 7）水平方向の連結
>> A = [1, 2, 3];
>> B = [4, 5, 6];
>> AA = DD(A);
>> BB = DD(B);
>> [AA, BB]
ans =
hi: [1 2 3 4 5 6]
lo: [0 0 0 0 0 0]
例 8）垂直方向の連結
>> A = [1, 2, 3];
>> B = [4, 5, 6];
>> AA = DD(A);
>> BB = DD(B);
>> CC = [AA ; BB];
>> CC.hi
ans =
1 2 3
4 5 6
>> CC.lo
ans =
0 0 0
0 0 0

3.9 MATLAB 関数との互換性

DD, QD 型でも MATLAB のいくつかの関数を使用することができます.

3.9.1 MATLAB 関数と同じように使える関数

・[m, n] = size(a)
a の行の長さ m と列の長さ n を返します.
・abs(a)
a の絶対値を返します.
・sqrt(a)
a の平方根を返します.
・dot(a, b)
a と b の内積を返します.
他の関数
・ddpow(a, n).
a を n 乗した値を返します.
・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.

4 並列処理

MuPAT で利用している並列処理の手法は 3 種類で, FMA, AVX2, OpenMP です．これらの並列処理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)，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自動的に指定されなかった機能が off になります.
n1 は非負数の入力が想定され，n2，n3 は 0 (off) か 1 (on) の入力が想定されています.
または,
>> fmaon
>> fmaoff
>> avxon
>> avxoff
>> omp(n1)
のようにして後から設定を変更することもできます．omp(n1) の引数としては非負数が想定されます.

4.1 FMA

FMA(Fused Multiply Add) 演算 [4, 5] は，x = a × b + c の形式で表される積和演算を 1 演算で実行できます．これにより，積和演算の演算回数を最大で半減でき，最大で 2 倍の性能向上が見込めます．ただし，FMA を用いて高速化できるのは積と和の組で表される場合のみです．
DD, QD 演算では FMA 向けのアルゴリズム (troprod_fma[2]) がサポートされており，MuPAT ではそのアルゴリズムを採用しています．そのため，FMA を使った場合と使わない場合で内部のアルゴリズムが異なります．

4.2 AVX2

Intel AVX2(Advanced Vector Extensions)[4, 5] は，1 命令で 4 つの倍精度浮動小数点数演算を実行できます．メモリアクセス数は変わらないため，最大 4 倍の性能向上が見込めます．
ただし，AVX2 を用いる際，入力の行列の次数が 4 の倍数でないとき，余った部分は AVX2 を使わず，逐次的に処理しています．

4.3 OpenMP

OpenMP[6] は共有メモリ型マシンで並列プログラミングを可能にする API で，コア数の分だけ高速化が可能です．スレッド数とは，並列処理をするときに分ける処理の単位を表しています．
また，MuPAT では環境変数を設定してスレッド数を変えることはできなくなっています．

4.4 高速化されている処理

MuPAT では上記の 3 種類の並列化を用いています．その結果，3 種類を全て併用した際には実行時間が短くなります．高速化の対象となった 48 種類の演算を以下の表に記します．

表 4.1 現在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	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							
行列ベクトル積	A: matrix, b: vector → A * b							
転値行列ベクトル積	A: matrix, b: vector → tmv(A, b)							
行列積	A, B: matrix → A * B							

※疎行列形式はサポートしていません．

例えば、intel core i7 7820 HQ 2.9 GHz を搭載した 4 コアマシンでは、 次数 4,192,000 の内積は DD-DD で 6.6 倍高速化され、約 0.005 秒で計算でき、QD-QD で 16.4 倍高速化され、約 0.016 秒で計算できます。次数 2,048 の行列ベクトル積では DD-DD で 7.9 倍高速化され、約 0.0034 秒で計算でき、QD-QD で 15.5 倍高速化され、約 0.016 秒で計算できます。さらに、次数 500 の行列積では DD-DD で 19.1 倍高速化され、約 0.043 秒で計算でき、QD-QD で 17.1 倍高速化され、約 0.46 秒で計算できるようになっています。

4.5 注意点

使用するマシンに対する，扱う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使っている PC のメモリサイズと比べてある程度大きいと MATLAB が強制終了してしまうことがあります．また，それは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ってしまい，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．

参考文献

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

[2] 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).

[3] 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).

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

[5] 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

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

[7] 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
  - @DD
    - DD.m
  - @QD
    - QD.m
  - @Calc
    - Calc.m
  - functions
    - useful
      - ddeye.m
      - ddrand.m
      - qdeye.m
      - qdrand.m
      - avxoff.m
      - avxon.m
      - fmaoff.m
      - fmaon.m
      - omp.m
    - maincode
      - mupat.c
      - mupat.h
      - mupat.o
      - set of C functions to build MEX file (48 files)
      - set of MEX files (48 files)