

**MATLAB 版**  
**MuPAT ユーザーガイド**  
version 1.0.0

八木武尊

2019 年 5 月 20 日

## 目次

1	はじめに	2
1.1	システム要件	2
2	MuPAT 起動方法	2
2.1	インストール	2
2.2	起動	2
3	使用方法	3
3.1	変数の定義	3
3.2	型の変換	4
3.3	入出力	5
3.4	算術演算	5
3.5	関係演算子	6
3.6	配列要素への代入と抽出	6
3.7	DD 型, QD 型で使える関数	7
3.8	Scilab 版 MuPAT との違い	8
4	並列処理	8
4.1	FMA	8
4.2	AVX2	8
4.3	OpenMP	8
4.4	並列化の効果が期待できる処理と表現	9
	参考文献	9
	付録 1: フォルダ構造	10

## 1 はじめに

MuPAT は Scilab[1] と, MATLAB[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. MuPAT では, Scilab または 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, 簡単に高精度演算を扱うことができます. さらに, MATLAB 版の MuPAT では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います. 現在, MATLAB 版は macOS のみに対応しています.

### 1.1 システム要件

macOS (10.13 High Sierra 以降)  
MATLAB R2017a 以降  
CPU Intel Haswell 以降, AMD は調査中

## 2 MuPAT 起動方法

### 2.1 インストール

MuPAT を Web ページ <https://www.ed.tus.ac.jp/1419521/> から zip 形式でダウンロードし, MuPAT ディレクトリを, MATLAB ディレクトリ下など, 作業しやすい場所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  

```
>> startMuPAT
```
4. MATLAB コマンドウィンドウで `a = DD(1)` と入力し, 次の結果が出力されれば MuPAT は正常に動作しています.  

```
>> a = DD(1)

a =
hi:  1
lo:  0
```

`startMuPAT` は入力引数としてスレッド数, AVX2 の on/off, FMA の on/off を持ちます. (引数なしはスレッド数 = 1 で, FMA, AVX2 はオフです). FMA, AVX2, OpenMP の説明については 4 節の並列処理を参照してください. 並列化を使用した際には計算順序が逐次のときと変わるため, 計算結果が一致しない場合があります.

例)

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

### 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 上の計算はすべて倍精度 (double) で行われます。MuPAT 上では、擬似 4 倍精度演算のために DD 型、擬似 8 倍精度演算のために QD 型という新しいデータ型を定義し、倍精度計算の組み合わせで高精度演算を実現しています。DD 型は 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 hh (最上位パート), hl, lh, ll (最下位パート) の 4 つの double 型変数を持ちます。DD 型変数 A の上位パート hi や下位パート lo はそれぞれ A.hi, A.lo, QD 型変数 B の最上位から最下位までは B.hh, B.hl, B.lh, B.ll で参照できます。

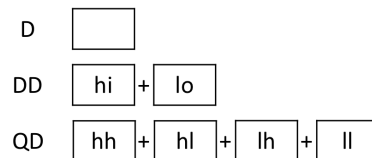


図 3.1 DD 型の数と QD 型の数のイメージ

DD 型の変数 A を定義するためには DD 関数を用いて以下のようにします。DD 関数は引数として倍精度の値または変数を 2 個とります。

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

- DD()  
上位パート hi と下位パート lo を 0 とします。
- DD(a)  
上位パート hi を a, 下位パート lo を 0 とします。
- DD(a, b)  
上位パート hi を a, 下位パート lo を b とします。

QD 型の変数 B を定義するためには QD 関数を用いて以下のようにします。QD 関数は引数としては倍精度の値または変数を 4 個とります。

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

- QD()  
最上位パート hh から最下位パート ll まですべて 0 とします。
- QD(a)

最上位パート hh を a, 他は 0 とします.

- QD(a, b)

最上位パート hh を a, hl を b, 他は 0 とします.

- QD(a, b, c)

最上位パート hh を a, hl を b, lh を c, 最下位パート ll は 0 とします.

- QD(a, b, c, d)

最上位パート hh を a, hl を b, lh を c, 最下位パート ll は d とします.

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], 1e-18 * [7, 8, 9; 10, 11, 12])
A =
hi:  [2x3 double]
lo:  [2x3 double]
>> A.hi
ans =
1 2 3
4 5 6
>> A.lo
ans =
1.0e-16 *
0.0700  0.0800  0.0900
0.1000  0.1100  0.1200
```

## 3.2 型の変換

double 型変数 a, DD 型変数 b, QD 型変数 c の型を相互に変換するときは以下のようにします. スカラー, ベクトル, 行列の型が代入の左辺と左辺で異なる場合の操作に関しては 3.8 節を確認ください.

表 3.1 型の変換方法

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

例)

```
>> a = double(b)
```

$$\gg c = DD(c)$$

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。

- `qdinput(s)`  
文字で表された数値列 `s` を QD 型の数値へ変換します。

四則演算子 (+, -, \*, /) はすべての型に使用できます。除算はスカラーのみ対応しています。型の異なる演算の結果は精度が高い方の型で保存されます。また、ドット演算 (.\*, ./) も使用できます。

```
>> d = 1 + 1.0e-20 % 倍精度加算
d =
1
>> e = DD(1, 1.0e-20) % DD 型の数
e =
hi: 1
lo: 1.0000e-20
>> e - d % 混合精度演算
ans =
hi: 1.0000e-20
lo: 0
```

```
'3.3333333333333333333333333333333333333333333333333333333E-1'
```

※コマンドウィンドウが小さいと、出力が省略される場合があります。

### 3.5 関係演算子

型が異なる場合でも、DD 型、QD 型で論理演算子 ( $=, \sim=, <, >, \leq, \geq$ ) や論理演算子 ( $\&, |, \sim$ ) が使用できます。

例)

```
>> a = 1/7;
>> b = 1/DD(7);
>> a == b
ans =
logical
0
>> a == b.hi % 上位桁は一致
ans =
logical
1
>> a < b % double の数の下位桁は 0
ans =
logical
1
```

### 3.6 配列要素への代入と抽出

配列 A の (i, j) 要素に数値もしくは変数 b を代入する際は、左辺の精度は右辺と同じか、それより高精度である必要があります。そうでない場合はエラーになります。

例)

```
>> A = DD([1,2,3; 4,5,6], 1e-18 * [7,8,9;10,11,12]);
>> A(2, 3)
ans =
hi: 6
lo: 1.2000e-17
>> A(2, 2) = 10;
>> A.hi
ans =
1 2 3
4 10 6
>> A.lo
ans =
1.0e-16 *
0.0700 0.0800 0.0900
0.1000 0 0.1200
>> A(2, 2) = QD(1); % A が DD 型配列なので QD 型要素を代入できずにエラー
```

error:

You should cast DD to QD.

### 3.7 DD 型, QD 型で使える関数

擬似乱数は, DD 型では上位パート, 下位パートそれぞれで MATLAB 関数 `rand()` によって乱数を作り, 上下のパートを足して 1 つの値となるように下位パートの値を調節しています. QD 型も同様に最上位から最下位までそれぞれ MATLAB 関数 `rand()` によって乱数を作り, 1 つの値となるように最上位以外のパートの値を調節しています.

- `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 と B の型が違う場合はキャストされた結果が返ります).
- `[A; B]`  
A, B を垂直方向に連結します (A と B の型が違う場合はキャストされた結果が返ります).
- `[m, n] = size(a)`  
行列 a の行の長さ m と列の長さ n を返します.
- `ddrand(m, n)`  
m 行 n 列の DD 型擬似乱数行列を返します.
- `qdrand(m, n)`  
m 行 n 列の QD 型擬似乱数行列を返します.
- `ddeye(m, n)`  
m 行 n 列の DD 型単位行列を返します.
- `qdeye(m, n)`  
m 行 n 列の QD 型単位行列を返します.
- `a^n`  
スカラー a を n 乗した値を返します.
- `abs(a)`  
変数 a の絶対値を返します.
- `sqrt(a)`  
変数 a の平方根を返します.
- `norm(a, n)`  
a の n ノルムを返します. n は 1, 2, inf, fro が選べます. (a がベクトルの場合のみ n = 2 を選べます)
- `dot(x, y)`  
ベクトル x と y のドット積を返します.
- `tmv(A, b)`  
行列 A' とベクトル b の転置行列ベクトル積を返します. (A' \* x でも計算できます)

DD 型, QD 型をサポートしていない MATLAB 関数に DD 型の数, QD 型の数を与えてを呼び出すと以下のよう  
なエラーになります.

例)

```
>> a=ddrand(5);
```

```
>> qr(a)
```



Undefined function 'qr' for input arguments of type 'DD'.

### 3.8 Scilab 版 MuPAT との違い

DD 型, QD 型を宣言する際, Scilab 版では小文字で `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. Scilab 版で型を変換する際には, 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()` 関数が必要でしたが, MATLAB 版では `DD()`, `QD()` などの変数の宣言と同じ関数を用います. 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, `startMuPAT` コマンドを実行します.

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

引数 `n1` は OpenMP のスレッド数 (off はスレッド数=1), `n2` は AVX2 の on は 1, off は 0, `n3` は FMA の on は 1, off は 0 を指定します. 指定しなかった場合は off となります. なお, 並列化を利用した際には計算順序が変わるため, 逐次のときと計算結果が変わります.

以下のようにして並列処理の on/off を切り替えることもできます.

```
>> fmaon
>> fmaoff
>> avxon
>> avxoff
>> omp(n1) % n1 でスレッド数を指定
```

現在の設定状況を確認するには `statusMuPAT` コマンドを用います.

```
>> statusMuPAT
```

### 4.1 FMA

FMA(Fused Multiply Add) 演算 [5, 6] は,  $x = a \times b + c$  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. FMA を有効にすると FMA 向けの計算アルゴリズム (`twoprod_fma`[4]) を使用するため, 積演算のアルゴリズムが異なります.

### 4.2 AVX2

Intel AVX2(Advanced Vector Extensions)[5, 6] では, 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.

### 4.3 OpenMP

OpenMP[7] を利用することで, プログラムがマルチスレッドで実行され, 最大コア数の分だけ性能向上が見込めます. スレッド数はユーザーの指定によって決まり, コア数より多く指定してもあまり効果はありません.

#### 4.4 並列化の効果が期待できる処理と表現

6 種類の演算 (ベクトル和, スカラー倍, 内積, 行列ベクトル積, 転置行列ベクトル積, 行列積) において並列実行を可能にしています. なお, `double` 型どうしの組み合わせを除いた 8 種類の型の組み合わせ (`double` と `DD`, `double` と `QD`, `DD` と `double`, `DD` と `DD`, `DD` と `QD`, `QD` と `double`, `QD` と `DD`, `QD` と `QD`) において通常と同じ操作で上記の演算を実行できます.

#### 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. Jan. 2013.
- [2] 八木武尊, 石渡恵美子, 長谷川秀彦, “並列処理を用いた対話的多倍長演算環境 MuPAT の高速化,” 第 17 回情報科学技術フォーラム, 第 1 分冊, pp.43-48. Sep. 2018.
- [3] D. H. Bailey, “High-Precision Floating-point arithmetic in scientific computation”, *Computing in Science and Engineering*, Vol.7, no.3, pp.54-61. May 2005.
- [4] Y. Hida, X. S. Li, and D. H. Baily, “Quad-double arithmetic: algorithms, implementation and application,” *Technical Report LBNL-46996*, Oct. 2000.
- [5] Intel, Intrinsics Guide: <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.

