CBZ/GroupDataの使用マニュアル

千葉豪

平成29年12月18日

数値計算においては、行列、ベクトルの演算が大部分を占めると言っても過言ではない。

様々な便利なツールが世界中の人によって開発、公開されている昨今、行列、ベクトル演算のための便利な数値計算ライブラリも入手可能となっている。しかし、「極力、ブラックボックスで便利なツールを使わない」という方針のもと、CBZにも、性能は悪いが、行列、ベクトルの演算を行うためのクラスGroupDataが実装されている。

GroupDataクラスは、一次元のベクター配列とその大きさをメンバー変数に持っており、それを継承する形で、ベクトルのためのクラスGroupData1Dと、行列のためのクラスGroupData2Dが実装されている。

1 基本的な使い方

ベクトルを定義するためのクラスGroupData1Dに関する基本的な使用例を以下に示す。

Listing 1: GroupData1Dの基本的な使用例

```
1  GroupData1D a(2);
2  a.put_data(0,2.);
3  a.put_data(1,-3.);
4
5  GroupData1D b;
6  b.put_imax(2);
7  b.put_data(0,5.);
8  b.put_data(1,4.);
9
10 GroupData1D c=a+b;
11 GroupData1D d=a-b;
12
13 real c0=a.get_dat(0)+b.get_dat(0);
14 real c1=a.get_dat(1)+b.get_dat(1);
15
16 real tmp=a*b;
17 GroupData1D e=a.mult(b);
18
19 e.show_self();
```

インスタンスの生成は、1行目にあるようにそのサイズを指定する方法と、5行目にあるように指定しない方法とがある。5行目のようにコンストラクタでベクトルのサイズを指定しない場合には、6行目にあるように別途「put_imax」メソッドによりサイズを指定する必要がある。

GroupData1Dへの数値データの入力は「put_data」メソッドを用いる。一つ目の引数がベクトルの位置（0から始まることに注意)、二つ目の引数がその位置に入力する数値データに対応する。また、GroupData1Dからの数値データの出力は「get_dat」メソッドを用いる。引数は、数値データを取り出すベクトルの位置に対応する。

なお、ベクトルのサイズは「get_imax」メソッドにより取り出す。

GroupData1D クラスのインスタンス同士の加算、減算は、10、11 行目で示すような形式で実行可能である。また、演算子「*」はベクトル間の内積計算を行うため、戻り値は実数となる。ベクトルの要素同士の積からなるベクトルを計算する場合には、17 行目にあるように「mult」メソッドを用いる。なお、これらのベクトル同士の演算を行う場合には、ベクトルのサイズが一致している必要がある。

GroupData1D クラスのデータを出力させる場合には、19 行目のように「show_self」メソッドを用いる。

次に、行列を定義するためのクラス GroupData2D に関する基本的な使用例を以下に示す。

Listing 2: GroupData2D の基本的な使用例

```
1  GroupData2D a(2,2);
2  a.put_data(0,0,2.);
3  a.put_data(0,1,5.);
4  a.put_data(1,0,-3.);
5  a.put_data(1,1,-3.);
6
7  GroupData2D b;
8  b.put_yx(2,2);
9  b.put_data(0,0,5.);
10 b.put_data(0,1,4.);
11 b.put_data(1,0,-1.);
12 b.put_data(1,1,0.);
13
14 GroupData2D c=a+b;
15 GroupData2D d=a-b;
16
17 real c00=a.get_dat(0,0)+b.get_dat(0,0);
18 real c01=a.get_dat(0,1)+b.get_dat(0,1);
19 real c02=a.get_dat(1,0)+b.get_dat(1,0);
20 real c03=a.get_dat(1,1)+b.get_dat(1,1);
21
22 GroupData2D e=a*b;
23
24 e.show_self();
```

基本的には GroupData1D の使い方と同様である。「get_dat」メソッドにより数値データを取り出すが、例えば「get_dat(y,x);」とした場合には第 y 行、第 x 列のデータが取り出されることになる。また、演算子「*」は通常の行列の積の計算を示しており、要素毎の積とはなっていないことに注意されたい。

いくつかの部分行列で構成される大きな行列を GroupData2D クラスで構築する場合、部分行列に対応する GroupData2D クラスのインスタンスを用いて、大きな行列の GroupData2D クラスのインスタンスに値を代入することができる。このためのメソッド「PasteGroupData2D」の使用例を以下に示す。

Listing 3: GroupData2D データの他の GroupData2D データへの上書き例

```
1  GroupData2D a(2,2);
2  a.put_data(0,0,2.);
3  a.put_data(0,1,-5.);
4  a.put_data(1,0,5.);
5  a.put_data(1,1,-3.);
6
7  GroupData2D b(5,5);
8  b.set_zero();
9  b.PasteGroupData2D(1,2,a);
```

この例では、2x2 の行列である変数「a」を、5x5 の行列「b」の (1,2) の位置（「1」が行、「2」が列に対応する）を基準にして上書きする操作を行っている。この例の出力結果を以下に示す。

Listing 4: GroupData2D データの他の GroupData2D データへの上書き例の出力結果

```
x:5 & y:5
*0*0.000000e+00:0.000000e+00:0.000000e+00:0.000000e+00:0.000000e+00:
*1*0.000000e+00:0.000000e+00:2.000000e+00:-5.000000e+00:0.000000e+00:
*2*0.000000e+00:0.000000e+00:5.000000e+00:-3.000000e+00:0.000000e+00:
*3*0.000000e+00:0.000000e+00:0.000000e+00:0.000000e+00:0.000000e+00:
*4*0.000000e+00:0.000000e+00:0.000000e+00:0.000000e+00:0.000000e+00:
```

同様に、GroupData2D クラスのインスタンスに GroupData1D クラスのインスタンスの情報を上書きするメソッド「PasteGroupData1D」も実装されている。その使用例を以下に示す。

Listing 5: GroupData1D データの GroupData2D データへの上書き例

```
  GroupData1D a(3);
  a.put_data(0,2.);
  a.put_data(1,-5.);
  a.put_data(2,3.);

  GroupData2D b(5,5);
  b.set_zero();
  b.PasteGroupData1D(1,2,a);
```

この例の出力結果を以下に示す。

Listing 6: GroupData1D データの GroupData2D データへの上書き例の出力結果

```
x:5 & y:5
*0*0.000000e+00:0.000000e+00:0.000000e+00:0.000000e+00:0.000000e+00:
*1*0.000000e+00:0.000000e+00:2.000000e+00:0.000000e+00:0.000000e+00:
*2*0.000000e+00:0.000000e+00:-5.000000e+00:0.000000e+00:0.000000e+00:
*3*0.000000e+00:0.000000e+00:3.000000e+00:0.000000e+00:0.000000e+00:
*4*0.000000e+00:0.000000e+00:0.000000e+00:0.000000e+00:0.000000e+00:
```

また、GroupData1D、GroupData2D いずれについても、サイズの設定を、インスタンス生成時ではなく、生成後に行うことが可能である。その例を以下に示す。なお、GroupData2D クラスの「put_yx」メソッドでは、一つ目の引数で行列の行数を、二つ目の引数で列数を、それぞれ指定する。

Listing 7: GroupData1D 及び GroupData2D のインスタンスのサイズを後から設定する方法の例

```
  GroupData1D a;
  a.put_imax(3);

  GroupData2D b;
  b.put_yx(5,5);
```

2 応用

2.1 逆行列の計算

GroupData2D クラスのインスタンスが正方行列を定義している場合、その逆行列を計算することが出来る（内部ではガウスの消去法で計算している）。計算例を以下に示す。

Listing 8: GroupData2D を用いた逆行列の計算例

```
  GroupData2D a(2,2);
  a.put_data(0,0,2.);
  a.put_data(0,1,5.);
  a.put_data(1,0,-3.);
  a.put_data(1,1,-3.);

  a.show_self();

```

```
9   GroupData2D b=a.inverse();
10
11  b.show_self();
12
13  a.DoInverse();
14
15  a.show_self();
```

9行目に示されているように、「inverse」メソッドを用いた場合には、逆行列からなるGroupData2Dクラスのインスタンスが返される。一方、インスタンス自身を逆行列に変換する場合は、13行目のように「DoInverse」メソッドを用いればよい。

2.2 連立一次方程式の解の計算

連立一次方程式は以下のように行列とベクトル形式で記述される。この

$$\mathbf{Ax} = \mathbf{b} \tag{1}$$

A、**b**が与えられたときの**x**を計算する例を以下に示す。

Listing 9: 連立一次方程式の解の計算例

```
1   GroupData2D a(2,2);
2   a.put_data(0,0,2.);
3   a.put_data(0,1,5.);
4   a.put_data(1,0,-3.);
5   a.put_data(1,1,-3.);
6
7   GroupData1D b(2);
8   b.put_data(0,3.);
9   b.put_data(1,-1.);
10
11  a.solveaxb_mod(b);
12
13  b.show_self();
14
15  GroupData2D c(2,2);
16  c.put_data(0,0,2.);
17  c.put_data(0,1,5.);
18  c.put_data(1,0,-3.);
19  c.put_data(1,1,-3.);
20
21  a.solveaxb_mod(c);
22
23  c.show_self();
```

ここでは行列**A**がGroupData2Dクラスのインスタンス「a」に、ベクトル**b**がGroupData1Dクラスのインスタンス「b」に、それぞれ対応する。このとき、GroupData2Dクラスのメソッド「solveaxb_mod」を用いることにより、ベクトル**x**を計算することが出来る。このメソッドでは引数で与えられたGroupData1Dクラスのインスタンスに解**x**が上書きされることになる。

また、以下のような行列形式の方程式の解**X**も計算可能である。

$$\mathbf{AX} = \mathbf{C} \tag{2}$$

これは、上の例の後半部分に対応する。

なお、これらの計算は全てガウスの消去法により行われている。

2.3 行列指数の計算

燃焼計算や動特性計算においては、以下のような行列形式の方程式が用いられることがある。

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) \tag{3}$$

この方程式の解は形式的に以下のように与えられる。

$$\mathbf{x}(t) = \exp(\mathbf{A}t)\,\mathbf{x}(0) \tag{4}$$

この式に現れる $\exp(\mathbf{A}t)$ を「行列指数（Matrix exponential）」といい、この行列を数値的に計算する方法が GroupData2D にはいくつか実装されている。

行列指数を直接計算するいくつかの例を以下に示す。

Listing 10: GroupData2D を用いた行列指数の計算例

```
1  GroupData2D a(2,2);
2  a.put_data(0,0,2.);
3  a.put_data(0,1,5.);
4  a.put_data(1,0,-3.);
5  a.put_data(1,1,-3.);
6
7  GroupData2D b=a.CalMatrixExponentialByPade(1.);
8  b.show_self();
9
10 GroupData2D c=a.CalMatrixExponentialByChebyshev14(1.);
11 c.show_self();
12
13 GroupData2D d=a.CalMatrixExponentialByMMPA32(1.);
14 d.show_self();
15
16 GroupData2D e=a.CalMatrixExponentialByChebyshevNew14(1.);
17 e.show_self();
```

GroupData2D クラスは、それ自身に定数（式 (4) における「t」）が乗ぜられた行列の行列指数を計算するメソッドをいくつか有している。これらのメソッドに共通するのは、自身に乗ぜられる定数を引数として指定するという点である。上の例の 7 行目の「CalMatrixExponentialByPade」メソッドでは、[6/6] 次の Pade 近似により計算を行う [1]。10 行目の「CalMatrixExponentialByChebyshev14」メソッドでは、14 次のチェビシェフ有理多項式近似（Chebyshev rational approximation method, CRAM）により計算を行う [2]。13 行目の「CalMatrixExponentialByMMPA32」メソッドでは、32 次のミニマックス多項式近似（Mini-max polynomial approximation, MMPA）により計算を行う [3]。16 行目の「CalMatrixExponentialByChebyshevNew14」メソッドでは 14 次の CRAM を用いるが、そこで用いているパラメータは川本氏が新たに計算したものである [4]。

実際には、行列指数を直接計算するよりも、$\mathbf{x}(0)$ と $\mathbf{A}t$ から、ベクトル $\mathbf{x}(t)$ を計算するほうが計算コストは小さくて済む。式 (4) の解を直接計算するいくつかの例を以下に示す。

Listing 11: 燃焼および動特性方程式の解の計算例

```
1  GroupData2D a(2,2);
2  a.put_data(0,0,2.);
3  a.put_data(0,1,5.);
4  a.put_data(1,0,-3.);
5  a.put_data(1,1,-3.);
6
7  GroupData1D x(2);
8  x.put_data(0,1.);
9  x.put_data(1,1.);
10
```

```
11  GroupData1D x1=a.CalMatrixExponentialByKrylov(x,1.,2);
12  x1.show_self();
13
14  //GroupData1D x2=a.CalMatrixExponentialByChebyshev14(x,1.);
15  GroupData1D x2=a.CalMatrixExponentialByChebyshev16(x,1.);
16  x2.show_self();
17
18  //GroupData1D x3=a.CalMatrixExponentialByMMPA18(x,1.);
19  GroupData1D x3=a.CalMatrixExponentialByMMPA32(x,1.);
20  x3.show_self();
21
22  GroupData1D x4=a.CalMatrixExponentialByChebyshevNew14(x,1.);
23  x4.show_self();
```

この場合、$\mathbf{x}(0)$ にあたる「初期ベクトル」を引数の一つ目として渡す必要がある。引数の二つ目は定数「t」に該当するものである。なお、この例では「CalMatrixExponentialByKrylov」メソッドが 11 行目に使われているが、これはクリロフ部分空間法を用いた解法が採用されている [1]。三つ目の引数は部分空間の次元数に対応する。

2.4 正方行列の対角化

GroupData2D クラスには、正方行列の対角化を行う機能が実装されている。正方行列を $\mathbf{A}$ としたとき、その対角化は以下の式で記述される。

$$\mathbf{AX} = \mathbf{XD} \tag{5}$$

ここで $\mathbf{D}$ は対角行列であり、各々の対角要素が行列 $\mathbf{A}$ の固有値となっている。CBZ ではこの対角化を古典的な方法である「Jacobi reduction」により計算している [1]。以下に GroupData2D クラスの対角化計算機能を用いた例を示す。

Listing 12: GroupData2D を用いた行列の対角化の計算例

```
1   GroupData2D a(2,2);
2   a.put_data(0,0,2.);
3   a.put_data(0,1,5.);
4   a.put_data(1,0,5.);
5   a.put_data(1,1,-3.);
6   a.show_self();
7
8   GroupData2D b,c;
9   a.Diagonalization(b,c);
10
11  b.show_self();
12  c.show_self();
```

行列の対角化は「Diagonalization」メソッドを用いる。このメソッドでは、固有値で構成される対角行列 $\mathbf{D}$、固有ベクトルで構成される行列 $\mathbf{X}$ が計算され、引数として渡される。従って、予めそれらのための GroupData2D クラスのインスタンスを作成しておき（この例では、b、c が該当）、それらを Diagonalization メソッドの引数として渡している。

参考文献

[1] A. Yamamoto, “Numerical solution of stiff burnup equation with short half lived nuclides by the Krylov subspace method,” *J. Nucl. Sci. Technol.*, **44**, p.147-154 (2007).

[1] 具体的な方法については C.D.Mayer 著「Matrix analysis and applied linear algebra」の p.353 を参照のこと。本来はもっと洗練された方法を実装したいが、未着手である（もちろん、どこかからとってきても良いわけであるが、、、）。

[2] M. Pusa, J. Leppanen, “Computing the matrix exponential in burnup calculations,” *Nucl. Sci. Eng.,*, **164**, p.140-150 (2010).

[3] Y. Kawamoto, *et al.,* “Numerical solution of matrix exponential in burn-up equation using mini-max polynomial approximation,” *Ann. Nucl. Energy*, **80**, p.219-224 (2015).

[4] 川本、「軽水炉における崩壊熱・核種生成量の評価精度に関する研究」、北海道大学平成 26 年度修士論文.