CBZ/GroupDataの使用マニュアル[1]

2021/11/12　千葉　豪

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、既存の便利なツールをブラックボックス的に使わない」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。なお、CBZ パッケージに GroupDataTest というディレクトリが含まれており、そこに以下で紹介するサンプルプログラムが格納されている。

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、そのサイズを指定する方法（1 行目に対応）と指定しない方法（5 行目に対応）とがある。後者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には 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意味するため、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 クラスのインスタンスに定義されている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.);
```

[1] Document/CBG_Manual/GroupData

```
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();
```

基本的な使い方は GroupData1D クラスと同様である。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。数値データを代入する put_data メソッドも同様であり、put_data(y,x,val); とすることで、第 y 行、第 x 列に数値 val が代入される。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。

GroupData2D クラスのインスタンスに値を代入する場合、個々の要素に対して put_data メソッドを利用すると、行列のサイズが大きい場合には煩雑となってしまう。そこで以下のように配列データを直接 GroupData2D クラスに渡すことも可能である。

Listing 3: 配列データを用いた GroupData2D のインスタンスへの値の代入

```
1   GroupData2D a(2,2);
2   real ainp[]={
3     2., 5.,
4     -3., -3.
5   };
6   a.put_data(ainp);
```

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 4: 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 5: GroupData2D データの他の GroupData2D データへの上書き例の出力結果

```
1  x:5 & y:5
2  *0*0.000000e+00:0.000000e+00:0.000000e+00:0.000000e+00:0.000000e+00:
3  *1*0.000000e+00:0.000000e+00:2.000000e+00:-5.000000e+00:0.000000e+00:
4  *2*0.000000e+00:0.000000e+00:5.000000e+00:-3.000000e+00:0.000000e+00:
5  *3*0.000000e+00:0.000000e+00:0.000000e+00:0.000000e+00:0.000000e+00:
6  *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 6: GroupData1D データの GroupData2D データへの上書き例

```
1   GroupData1D a(3);
2   a.put_data(0,2.);
3   a.put_data(1,-5.);
4   a.put_data(2,3.);
5
6   GroupData2D b(5,5);
7   b.set_zero();
8   b.PasteGroupData1D(1,2,a);
```

この例の出力結果を以下に示す。

Listing 7: 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 のいずれについても、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 8: 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

```
1  GroupData1D a;
2  a.put_imax(3);
3
4  GroupData2D b;
5  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 9: GroupData2D を用いた逆行列の計算例

```
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  a.show_self();
8
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。

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

この式における $\mathbf{A}$、$\mathbf{b}$ が与えられたときの $\mathbf{x}$ を計算する例を以下に示す。

Listing 10: 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 $\mathbf{A}$ が GroupData2D クラスのインスタンス a に、ベクトル $\mathbf{b}$ 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 $\mathbf{x}$ 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 $\mathbf{x}$ 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 $\mathbf{X}$ 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 11: 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、それ自身になんらかの定数（例えば式 (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。以下、それぞれのメソッドについて説明を行う。

- 7 行目の CalMatrixExponentialByPade メソッドでは、[6/6] 次の Pade 近似 [1] により計算を行う。
- 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）[2] により計算を行う。

- 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）[3] により計算を行う。
- 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 [4] である。

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 12: 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 [1] が採用されている。三つ目の引数は部分空間の次元数に対応する。

2.4 対称行列の対角化

`GroupData2D` クラスには、対称行列の対角化を行う機能が実装されている。対称行列を $\mathbf{A}$ 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 [2]。以下に `GroupData2D` クラスの対角化計算機能を用いた例を示す。

Listing 13: `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。

[2] 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、、、）。

2.5 特異値分解

GroupData2D クラスのインスタンスに対して特異値分解を行うことが可能である。この計算では、対象とする行列 $\mathbf{A}$ について、$\mathbf{A}\mathbf{A}^T$ と $\mathbf{A}^T\mathbf{A}$ を計算し、これらの対称行列に対して Jacobi reduction により対角化を行い、最終的に $\mathbf{A}$ の特異値と特異値ベクトルを得る [3] 。Jacobi reduction による対角化の計算精度はそれほど高いものではないため、このような方法で特異値分解を行う場合には、その結果の精度に注意が必要である。

この計算には SVDTool クラスを用いる。SVDTool クラスを用いた特異値分解の例を以下に示す。

Listing 14: SVDTool を用いた行列の特異値分解の例

```
1  #include "Numeric.h"
2
3  #include "SVDTool.h"
4
5    int main()
6  {
7    int m=3;
8    int n=3;
9
10   real data_org[]={
11     1., 0., 0.,
12     1., -1., 2.,
13     1., 1., -2.,
14   };
15
16   GroupData2D a(m,n);
17   a.put_data(data_org);
18   //a.show_self();
19
20   SVDTool svd;
21
22   real sv_cond=1e-10;
23   // [Minimum value for singular value criteria]
24   //    If calculated singular value is smaller than this quantity,
25   //    it is regarded as NOT singular value BUT numerical rounding error.
26
27   real print_option=true;
28   // [Print option]
29   //    If this option is [true], singular values are printed on screen.
30   //    Singular values are calculated from eigenvalues of two matrices, AA^T and A^T A,
31   //    so a pair of singular values is printed.
32   //    Theoretically these two values should be indentical.
33   //    If not, it suggests that numerical calculation does NOT work well.
34   //    In such cases, please ask Chiba.
35
36   svd.SVD(a);
37   vector<real> sv=svd.GetSingularValue(sv_cond,print_option);
38
39   GroupData2D umat(m,n);
40   GroupData2D vmat(m,n);
41   for(int i=0;i<m;i++){
42     for(int j=0;j<n;j++){
43       umat.put_data(i,j,svd.GetUdata(j,i));
44       vmat.put_data(i,j,svd.GetVdata(j,i));
45     };
46   };
47
48   cout<<"#\n#_U_matrix\n#\n";
49   umat.show_self();
50
51   cout<<"#\n#_V_matrix\n#\n";
52   vmat.show_self();
53
54   //svd.MatrixReconstructionCheck(3);
55
56   return 0;
57 }
```

36 行目にあるように、SVDTool クラスのメソッド SVD により、引数として渡される GroupData2D クラスのインスタンスとして定義される行列に対して特異値分解を行う。37 行目の GetSingularValue メソッドでは、特異値分解の結果得られた特異値を vector 型の配列として取り出している。ここで、1 つ目の引数 sv_cond では有意と見做す特異値の閾値を設定し、2 つ目の引数 print_option では、計算結果の出力オプションを指定する。以下に、GetSingularValue メソッドの出力例を示す。

Listing 15: SVDTool クラスの GetSingularValue メソッドの出力例

```
1  # Warning in MatrixDiagonalization.
2  # Matrix becomes diagonal at iteration 1
3  # The size of matrix is 3
```

[3] 行列 $\mathbf{A}$ を $\mathbf{A}=\mathbf{U}\mathbf{D}\mathbf{V}^T$ のように特異値分解できるとする。この両辺について転置をとると、$\mathbf{A}^T=\mathbf{V}\mathbf{D}^T\mathbf{U}^T$ と出来る。従って、$\mathbf{A}\mathbf{A}^T=\mathbf{U}\mathbf{D}\mathbf{V}^T\mathbf{V}\mathbf{D}^T\mathbf{U}^T=\mathbf{U}\mathbf{D}\mathbf{D}^T\mathbf{U}^T=\mathbf{U}\mathbf{D}^2\mathbf{U}^T$ と $\mathbf{A}^T\mathbf{A}=\mathbf{V}\mathbf{D}\mathbf{U}^T\mathbf{U}\mathbf{D}^T\mathbf{V}^T=\mathbf{V}\mathbf{D}\mathbf{D}^T\mathbf{V}^T=\mathbf{V}\mathbf{D}^2\mathbf{V}^T$ が得られ、$\mathbf{A}\mathbf{A}^T$ と $\mathbf{A}^T\mathbf{A}$ の固有値から $\mathbf{A}$ の特異値と特異ベクトルを計算出来ることが分かる。

```
# The number of non-zero elements is 5
#
# Singular value from M (AA^T) and N (A^T A)
#   (Imaginary values are regarded as zero.)
#   (If two values are different from each other,
#    you need to be careful about the results.)
#
0     3.1622776602e+00 3.1622776602e+00
1     1.7320508076e+00 1.7320508076e+00
```

SVDTool クラスが行う特異値分解は、前述の通り、$\mathbf{A}\mathbf{A}^T$、$\mathbf{A}^T\mathbf{A}$ の固有値分解により行う。より具体的には、これらの行列それぞれの固有値から特異値が計算され、それらが 2 つ目の引数を true にすることによって出力される。理論的には、それぞれの行列から計算される特異値は一致しなければならないと言えるが、採用しているアルゴリズムの数値計算精度が原因で両者が有意に異なる場合があり、そのような場合には、計算結果をそのまま使用するのは避けたほうがよいと言える。その点を確認する観点から、この出力オプションを適切に利用することが重要である。

また、Listing 14 において、SVD メソッドにより特異値分解を行った後に、SVDTool のインスタンスから左特異ベクトル $\mathbf{u}$ と右特異ベクトル $\mathbf{v}$ を取り出すことが出来る。このために用いるのが GetUdata、GetVdata メソッドである。これらの引数は、1 つ目が特異ベクトルの Index を示しており、最も大きい特異値に対応するベクトルを取り出したい場合には 0 を、次に大きいものの場合は 1 を用いる。また、2 つ目が特異（列）ベクトルの行を示す。Listing 14 の例では、特異ベクトルが列ベクトルを構成する行列 $\mathbf{U}$、$\mathbf{V}$ を再構築しており、i、j のインデクスの使い方が若干わかりづらくなっている。

特異値分解を行った後に、一部の特異ベクトルを用いて再構築した行列の、オリジナルの行列に対する差異を計算するメソッド MatrixReconstructionCheck も実装されており、Listing 14 の最後にコメントアウトされている。このメソッドでは、引数において、行列を再構築する際に用いる基底ベクトルの個数（次元数）を入力する。出力例を以下に示す。行列の全要素について、Org. data でオリジナルの行列における値、Abs. Diff. to Ref. で再構築した行列での値のオリジナルの値に対する絶対差、Ratio to Ref. で再構築した行列での値のオリジナルの値に対する比を示す。行列のサイズが大きくなると膨大な情報が出力されることになるので、適当に GroupData クラスのメソッドを改変してもらいたい。

Listing 16: SVDTool クラスの MatrixReconstructionCheck メソッドの出力例

```
# Matrix reconstruction check
#
# The number of basis vectors (dimension) : 3
#
# y=0, x=0 : Org. data = 1 / Abs. Diff. to Ref. = 0 / Ratio to Ref. = 1
# y=1, x=0 : Org. data = 1 / Abs. Diff. to Ref. = -1.11022e-16 / Ratio to Ref. = 1
# y=2, x=0 : Org. data = 1 / Abs. Diff. to Ref. = 0 / Ratio to Ref. = 1
# y=0, x=1 : Org. data = 0 / Abs. Diff. to Ref. = -1.79439e-17 /
# y=1, x=1 : Org. data = -1 / Abs. Diff. to Ref. = 0 / Ratio to Ref. = 1
# y=2, x=1 : Org. data = 1 / Abs. Diff. to Ref. = 0 / Ratio to Ref. = 1
# y=0, x=2 : Org. data = 0 / Abs. Diff. to Ref. = 3.58878e-17 /
# y=1, x=2 : Org. data = 2 / Abs. Diff. to Ref. = 0 / Ratio to Ref. = 1
# y=2, x=2 : Org. data = -2 / Abs. Diff. to Ref. = 0 / Ratio to Ref. = 1
```

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).

[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.

A 部分行列について行列指数が計算できるときの同次方程式の解法とそのためのツール

2.3 節で述べたように、微分方程式

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) \tag{6}$$

の解は形式的に以下のように与えられる。

$$\mathbf{x}(t) = \exp(\mathbf{A}t)\,\mathbf{x}(0) \tag{7}$$

行列指数 $\exp(\mathbf{A}t)$ が数値的に計算できる場合は解が求められるが、そうではない場合は、以下の方法を用いることが出来る。

式 (6) を以下のように 2 つに分割して記述する。

$$\frac{d}{dt}\begin{pmatrix}\mathbf{x}_1(t)\\ \mathbf{x}_2(t)\end{pmatrix} = \begin{pmatrix}\mathbf{A}_{11} & \mathbf{A}_{12}\\ \mathbf{A}_{21} & \mathbf{A}_{22}\end{pmatrix}\begin{pmatrix}\mathbf{x}_1(t)\\ \mathbf{x}_2(t)\end{pmatrix} \tag{8}$$

ここで、行列指数 $\exp(\mathbf{A}_{22}t)$ は数値的に計算可能であるものとする。また、ベクトル $\mathbf{x}$ のサイズを I とし、$\mathbf{x}_1$、$\mathbf{x}_2$ のサイズをそれぞれ I_1、I_2 とする（$I = I_1 + I_2$）。

式 (8) は陽には以下で記述される。

$$\frac{d\mathbf{x}_1(t)}{dt} = \mathbf{A}_{11}\mathbf{x}_1(t) + \mathbf{A}_{12}\mathbf{x}_2(t), \tag{9}$$

$$\frac{d\mathbf{x}_2(t)}{dt} = \mathbf{A}_{21}\mathbf{x}_1(t) + \mathbf{A}_{22}\mathbf{x}_2(t) \tag{10}$$

式 (10) の両辺に左から $\exp(-\mathbf{A}_{22}t)$ を作用させると、$\exp(-\mathbf{A}_{22}t)\mathbf{A}_{22} = \mathbf{A}_{22}\exp(-\mathbf{A}_{22}t)$ より以下を得る。

$$\frac{d}{dt}\left(\exp(-\mathbf{A}_{22}t)\mathbf{x}_2(t)\right) = \exp(-\mathbf{A}_{22}t)\mathbf{A}_{21}\mathbf{x}_1(t) \tag{11}$$

式 (11) を $[t, t+\Delta t]$ で積分すると以下を得る。

$$\exp(-\mathbf{A}_{22}(t+\Delta t))\mathbf{x}_2(t+\Delta t) - \exp(-\mathbf{A}_{22}(t))\mathbf{x}_2(t) = \int_t^{t+\Delta t}\exp(-\mathbf{A}_{22}t')\mathbf{A}_{21}\mathbf{x}_1(t')dt' \tag{12}$$

この式に左から $\exp(\mathbf{A}_{22}(t+\Delta t))$ を作用させると、$\exp(\mathbf{A})\exp(\mathbf{B}) = \exp(\mathbf{A}+\mathbf{B})$ より以下が得られる。

$$\mathbf{x}_2(t+\Delta t) = \exp(\mathbf{A}_{22}\Delta t)\mathbf{x}_2(t) + \int_t^{t+\Delta t}\exp(\mathbf{A}_{22}(t-t'+\Delta t))\mathbf{A}_{21}\mathbf{x}_1(t')dt' \tag{13}$$

これに対して $\hat{t} = t' - t$ とおくと、$d\hat{t} = dt'$ より、上式右辺第二項の積分は

$$\int_0^{\Delta t}\exp(\mathbf{A}_{22}(\Delta t - \hat{t}))\mathbf{A}_{21}\mathbf{x}_1(\hat{t})d\hat{t} \tag{14}$$

と書き直せる。ここで、$\mathbf{x}_1(t)$ について、この区間では以下のように線形に変化するものと仮定する。

$$\mathbf{x}_1(\hat{t}) = \mathbf{x}_1(t) + \left(\mathbf{x}_1(t+\Delta t) - \mathbf{x}_1(t)\right)\frac{\hat{t}}{\Delta t} \tag{15}$$

これを式 (14) に代入すると以下を得る。

$$\exp(\mathbf{A}_{22}\Delta t)\left[\left(\int_0^{\Delta t}\exp(-\mathbf{A}_{22}\hat{t})d\hat{t}\right)\mathbf{A}_{21}\mathbf{x}_1(t) + \frac{1}{\Delta t}\left(\int_0^{\Delta t}\hat{t}\exp(-\mathbf{A}_{22}\hat{t})d\hat{t}\right)\left\{\mathbf{A}_{21}\left(\mathbf{x}_1(t+\Delta t) - \mathbf{x}_1(t)\right)\right\}\right] \tag{16}$$

式 (16) 中の積分については以下のように計算される。

$$\int_0^{\Delta t}\exp(-\mathbf{A}_{22}\hat{t})d\hat{t} = -\mathbf{A}_{22}^{-1}\left(\exp(-\mathbf{A}_{22}\Delta t) - \mathbf{I}\right), \tag{17}$$

$$\int_0^{\Delta t}\hat{t}\exp(-\mathbf{A}_{22}\hat{t})d\hat{t} = -\mathbf{A}_{22}^{-1}\Delta t\exp(-\mathbf{A}_{22}\Delta t) - \left(\mathbf{A}_{22}^{-1}\right)^2\left(\exp(-\mathbf{A}_{22}\Delta t) - \mathbf{I}\right) \tag{18}$$

ここで、行列 $\mathbf{I}$ は $I_2 \times I_2$ の単位行列である。これより、式 (16) は以下で記述される。

$$\exp(\mathbf{A}_{22}\Delta t)\left[-\mathbf{A}_{22}^{-1}\left(\exp(-\mathbf{A}_{22}\Delta t)-\mathbf{I}\right)\mathbf{A}_{21}\mathbf{x}_1(t)\right.$$
$$\left.+\left\{-\mathbf{A}_{22}^{-1}\exp(-\mathbf{A}_{22}\Delta t)-\frac{\left(\mathbf{A}_{22}^{-1}\right)^2}{\Delta t}\left(\exp(-\mathbf{A}_{22}\Delta t)-\mathbf{I}\right)\right\}\mathbf{A}_{21}\left(\mathbf{x}_1(t+\Delta t)-\mathbf{x}_1(t)\right)\right] \quad (19)$$

さらに、$\exp(\mathbf{A}t)\mathbf{A}^{-1}=\mathbf{A}^{-1}\exp(\mathbf{A}t)$ より、上式は以下のように書ける。

$$\mathbf{A}_{22}^{-1}\left\{\exp(\mathbf{A}_{22}\Delta t)+\frac{\mathbf{A}_{22}^{-1}}{\Delta t}\left(\mathbf{I}-\exp(\mathbf{A}_{22}\Delta t)\right)\right\}\mathbf{A}_{21}\mathbf{x}_1(t)+\mathbf{A}_{22}^{-1}\left\{-\mathbf{I}-\frac{\mathbf{A}_{22}^{-1}}{\Delta t}\left(\mathbf{I}-\exp(\mathbf{A}_{22}\Delta t)\right)\right\}\mathbf{A}_{21}\mathbf{x}_1(t+\Delta t) \quad (20)$$

従って、

$$\mathbf{M} = \exp(\mathbf{A}_{22}\Delta t), \quad (21)$$
$$\mathbf{E} = \mathbf{A}_{22}^{-1}\left\{\exp(\mathbf{A}_{22}\Delta t)+\frac{\mathbf{A}_{22}^{-1}}{\Delta t}\left(\mathbf{I}-\exp(\mathbf{A}_{22}\Delta t)\right)\right\}\mathbf{A}_{21}, \quad (22)$$
$$\mathbf{X} = \mathbf{A}_{22}^{-1}\left\{-\mathbf{I}-\frac{\mathbf{A}_{22}^{-1}}{\Delta t}\left(\mathbf{I}-\exp(\mathbf{A}_{22}\Delta t)\right)\right\}\mathbf{A}_{21} \quad (23)$$

とおくと式 (13) は以下のように書ける。

$$\mathbf{x}_2(t+\Delta t)=\mathbf{M}\mathbf{x}_2(t)+\mathbf{E}\mathbf{x}_1(t)+\mathbf{X}\mathbf{x}_1(t+\Delta t) \quad (24)$$

式 (9) を θ 法で差分化すると以下の式を得る。

$$\frac{\mathbf{x}_1(t+\Delta t)-\mathbf{x}(t)}{\Delta t}=\left\{\mathbf{A}_{11}\mathbf{x}_1(t+\Delta t)+\mathbf{A}_{12}\mathbf{x}_2(t+\Delta t)\right\}\theta+\left\{\mathbf{A}_{11}\mathbf{x}_1(t)+\mathbf{A}_{12}\mathbf{x}_2(t)\right\}(1-\theta) \quad (25)$$

この式を整理すると以下の式を得る。

$$\mathbf{N}_0\mathbf{x}_1(t+\Delta t)=\mathbf{N}_1\mathbf{x}_1(t)+\mathbf{N}_2\mathbf{x}_2(t) \quad (26)$$

ここで、

$$\mathbf{N}_0 = \mathbf{I}-(\mathbf{A}_{11}+\mathbf{A}_{12}\mathbf{X})\theta\Delta t, \quad (27)$$
$$\mathbf{N}_1 = \mathbf{I}+\mathbf{A}_{12}\mathbf{E}\theta\Delta t+\mathbf{A}_{11}(1-\theta)\Delta t, \quad (28)$$
$$\mathbf{N}_2 = \mathbf{A}_{12}\mathbf{M}\theta\Delta t+\mathbf{A}_{12}(1-\theta)\Delta t \quad (29)$$

であり、$\mathbf{I}$ は $I_1 \times I_1$ の単位行列である。これらの式より、$\mathbf{x}_1(t)$、$\mathbf{x}_2(t)$ が既知であれば $\mathbf{x}_1(t+\Delta t)$ が計算できることが分かる。また、$\mathbf{x}_2(t+\Delta t)$ については式 (24) を用いて計算できる。

`GroupData2D` クラスには、この方法に基づく SolveSimultaneousDifferentialEquationByDecomposedMatrixExponential メソッドが実装されている。このメソッドでは、行列 $\mathbf{A}$ に対して I_1、I_2、Δt、θ を指定することにより、行列 $\mathbf{M}$、$\mathbf{E}$、$\mathbf{X}$、$\mathbf{N}_0$、$\mathbf{N}_1$、$\mathbf{N}_2$ が計算される。