“KSHELL” User’s guide

清水 則孝
東京大学大学院 理学系研究科 附属原子核科学研究センター
e-mail: shimizu@cns.s.u-tokyo.ac.jp

平成 32 年 12 月 7 日

目 次

1 はじめに

原子核殻模型計算コード “KSHELL” は、M-scheme 基底を用い、Thick-restart Lanczos 法による殻模型計算をおこなうプログラムです。Linux PC 上で簡単に使え、OpenMP スレッド並列に対応しているので CPU コアを有効に使えます。さらに、同じ使い勝手で、MPI+OpenMP ハイブリッド並列による大規模並列計算にも使えることが特徴です。本説明書は、このプログラムの使い方を説明します。

このプログラムは簡便な対話型インターフェースを備え、ジョブ実行のためのシェルスクリプトを生成するので、このスクリプトのわずかな修正により、多くの計算機環境に対応できます。(Linux OS と Intel Fortran compiler が推奨です。) エネルギー準位、スピン、アイソスピン期待値、磁気能率、四重極能率、E2/M1/E1/GT 遷移確率、一粒子分光学的因子を求めることができます。M-scheme 次元にして、100 億次元を超える系の計算実績をもっており、^{56}Ni pf-shell KB3 相互作用の基底状態 (10 億次元の行列対角化に相当) を、145 秒で計算できました。(@東京大学 FX10 512 ノード、8192 コア)

2 使用許諾·免責条項

本ソフトウェアは学術目的にのみ使用を許可します。いかなる形においてもこのソフトウェアを利用する際は、下記の文献を引用してください。

N. Shimizu, T. Mizusaki, T. Utsuno, and Y. Tsunoda, Comp. Phys. Comm. **244**, 372 (2019) https://doi.org/10.1016/j.cpc.2019.06.011

N. Shimizu, “Nuclear shell-model code for massive parallel computation, KSHELL”, arXiv:1310.5431 [nucl-th] (2013).

本ソフトウェアは無保証です。本ソフトウエアを利用したことにより生じた損害、あるいは二次的に発生した一切の損害について作者は責任を負いません。

商用利用 · 再配布は原則的に禁止します。もし商用利用または再配布を希望する場合、個別に作者までご連絡ください。

情報提供、バグ報告、改良、相互作用 (snt) ファイルの提供などを歓迎します。e-mail にて作者 (shimizu@cns.s.u-tokyo.ac.jp) までお寄せください。

3 インストール

3.1 動作環境

必須動作環境

- Fortran compiler (Fortran 95, ISO-TR15581 拡張のサポートが必須)
- BLAS, LAPACK library
- python ver.2.4 or later (python ver.3 系列は未対応)

コンパイル時にオプションとして選択可能なもの

- C compiler (経過時間計測のみに使用)
- OpenMP によるスレッド並列
- MPI-2 library (+ MPI-IO) によるノード間並列

動作確認済環境：

- Intel Fortran 11.1 + Intel MKL, Linux OS
- GNU gfortran 4.6.3 + BLAS + LAPACK, Linux OS (gfortran ver.4.2 以降必須)
- PGI Fortran 13.10 + AMD-ACML ライブラリ, Linux OS
- 富士通 Fortran + SSLII, FX10/京コンピュータ
- gfortran, BLAS, LAPACK, cygwin on Windows 7
- Intel Fortran, Intel MKL, Mac OS

個人の環境で用いる場合は、新しめの Linux OS + Intel Fortran Compiler を推奨します。多くの場合、GNU gfortran でも問題ないでしょう。(gfortran の場合、lapack と blas のインストールが必要です。)

また、MS-Windows で用いる場合、cygwin 上で動かすのが最も簡単でしょう。Windows 7 + cygwin + gfortran で動作を確認しました。Mac OS X + Intel Fortran + OpenMP も問題なく動きました。(Makefile を編集する必要があります。)

3.2 インストール方法

インストールするディレクトリに移動。

```
tar xvzf kshell.tar.gz
cd kshell/src

Makefile を環境にあうよう編集 (デフォルトは Intel Fortran)

make single # for single node  (w/ or w/o OpenMP)
または、
make mpi    # for MPI parallel

alias kshell_ui.py=(installed directory)/bin/kshell_ui.py
```

これでインストールは終了です。単一ノード内の複数の CPU コアを用いたスレッド並列計算では、MPI parallel は不要です。“make single” としてください。自動的にコア数が判定され、スレッド並列に計算します。log ファイル先頭付近に

```
OpenMP  # of threads= 16
```

と表示されていれば、スレッド並列が有効になっています。

Makefile では、先頭部分の FC に適切なコンパイラコマンドを指定すれば、他のオプションは変更されるようになっていますが、環境によりさらなる修正が必要かも知れません。単一ノードと MPI 並列両方のバイナリをコンパイルする際には、make single && make clean && make mpi としてください。

3.3 ファイル構成

すべてのファイルは kshell ディレクトリ内の以下のディレクトリに格納されている。

- src ... ソースファイルを格納
- bin ... コンパイルされた実行ファイルを格納
  - bin/kshell_ui.py シェルスクリプト生成用対話型インターフェース
  - bin/gen_partition.py パーティションファイル生成用対話型インターフェース
  - bin/kshell ランチョス法による対角化計算用、バイナリ実行ファイル
  - bin/transit 遷移確率計算用、バイナリ実行ファイル
  - bin/count_dim パーティションファイルから M-scheme 次元を求めるバイナリ実行ファイル
- snt ... 相互作用ファイル (.snt) を格納。
  - snt/w.snt ... Wildenthal's USD 相互作用 (sd shell 用)
  - snt/gxpf1a.snt ... GXPF1A 相互作用 (pf shell 用)
  - その他いくつかの実績ある相互作用が用意されている。自作してここに保存することも可能。
- test ... テストディレクトリ（通常は使用しません。）

4 ジョブの生成・実行

流れとして、まず作業用ディレクトリを作成、移動します。kshell_ui.py を用いて、ジョブ用のシェルスクリプトを作成します。それを実行（単体ノード）あるいは、ジョブキューイングシステムに投入することによって計算をおこないます。

4.1 ジョブスクリプトの生成

新しいディレクトリを作成、移動して、kshell_ui.py と実行し、OXBASH ライクな対話型インターフェースを起動し、ジョブ用のスクリプトを生成します。

混乱を避けるため、常に空のディレクトリから始めることをおすすめします。

```
mkdir work     # 実行用に任意の作業用ディレクトリを作成
cd work
kshell_ui.py
....
```

以下のように、質問に答えていってください。snt ファイルの指定など、一部の入力にはタブキーによる補完が効きます。下記の例では、*sd* shell, USD 相互作用を用いて ^{24}Mg のエネルギー準位を 10 状態求めています。下線が引いてあるところが入力です。

```
kshell_ui.py

-----------------------------
  KSHELL user interface
     to generate job script.
-----------------------------


 MPI parallel?  Y/N (default:  No) :  n      MPI 並列の場合、y

  ... generate shell script for a single node.

 model space and interaction file name (.snt)

 (e.g.  w or w.snt, TAB key to complete) :  w.snt


*************** specify a nuclide ********************


 number of valence protons and neutrons

 (ex.  2, 3 <CR>) <CR> to quit :  4,4       アクティブな陽子・中性子数


 name for script file (default: Mg24_w ):

 J, parity, number of lowest states
  (ex. 10           for 10 +parity, 10 -parity states w/o J-proj. (default)
       -5           for lowest five -parity states,
       0+3, 2+1     for lowest three 0+ states and one 2+ states,
       1.5-, 3.5+3  for lowest one 3/2- states and three 7/2+ states) :

 +10       正パリティ状態を 10 状態求める


 truncation for "+" parity state in  Mg24_w_p.ptn
 truncation scheme ?
     (0): No trucation (default)
     (1): particle-hole truncation for orbit(s)
     (2): hw truncation
     (3): Both (1) and (2)
```

: 0　　トランケーションなし。トランケーションがある場合、細かく指定（後述）

```
generating partition file ............ done.

 --- input parameter ---
  beta_cm = 0.d0              # Lawson beta (MeV)
  eff_charge = 1.5, 0.5,      # effective charge
  fn_int = "w.snt"            # snt file
  gl = 1.0, 0.0,              # g-factor for orbital
  gs = 3.91, -2.678,          # g-factor for spin
  hw_type = 2                 # Harmonic oscillator parameter
  max_lanc_vec = 200          # iteration for Lanczos
  maxiter = 300               # iteration for TR-Lanczos
  mode_lv_hdd = 1             # Lanczos vector save in HDD or not
  n_restart_vec = 10          # restart vec for TR-Lanczos

modify parameter?
 (e.g.  maxiter = 300 for parameter change
        <CR>          for no more modification ) :
```

_　　ランチョス法の計算の詳細設定。変更なしの場合は、入力なしで Enter。
eff_charge = 1.8, 0.8　たとえば、有効電荷を指定。
各パラメータの詳細は Sect.5.3 を参照。

```
 compute transition probabilities (E2/M1/E1) for
    Mg24_w ? Y/N (default: No) :
```

y　　E2 遷移確率を求めるとき “y”。不要な時 “n”。

```
*************** specify a nuclide ********************

 number of valence protons and neutrons
  (ex.  2, 3 <CR>)    <CR> to quit :
```

: _　　Enter で終了。GT 遷移、分光学的因子を求めるためなどに、同時に他の核種を計算する場合は続けてアクティブな陽子・中性子数を指定して続ける。

```
 Finish. Execute ./Mg24_w.sh
```

Mg24_w.sh 他いくつかの必要ファイルが生成・コピーされます。単体ノード環境では、./Mg24_w.sh と実行、MPI 環境下では、 Mg24_w.sh を環境にあうよう修正し、ジョブを投入してください。

複数の核種も同一のシェルスクリプトに生成することができます。複数の核種の間のガモフテラー遷移、一粒子分光学的因子を求める場合、対話型インターフェースにて求めるか質問が表示されますので、“y” を選んでください。

4.2 トランケーションの設定

kshell_ui.py の実行途中でトランケーションについて入力します。kshell_ui.py は、パーティションファイルと呼ばれるトランケーションの情報をもったファイルを生成します (先の例では Mg24_w_p.ptn)。プログラム “kshell” は、パーティションファイルを読み取って M-scheme 空間を生成します。

前述の ^{24}Mg の例からすると、正パリティ状態しか存在しないので正パリティ状態のトランケーションの入力を促されます。(負パリティがある場合は続けて負パリティのトランケーションの入力を促されます。

```
truncation for "+" parity state in  Mg24_w_p.ptn
truncation scheme ?
     0 : No trucation (default)
     1 : particle-hole truncation for orbit(s)
     2 : hw truncation
     3 : Both (1) and (2)
```

0 を選べば、トランケーションなしで、許されるすべての M-scheme 配位を含みます。

1 は、(複数の) 軌道を選び、その占有粒子数の最大値、最小値を指定します。たとえば、0d5/2 軌道から 2 粒子までの励起を許すとすると、最小値 6 最大値 8 になるので以下のような入力です。

1

```
  #    n,  l,  j, tz,    spe
  1    0   2   3  -1     1.647
  2    0   2   5  -1    -3.948
  3    1   0   1  -1    -3.164
  4    0   2   3   1     1.647
  5    0   2   5   1    -3.948
  6    1   0   1   1    -3.164
specify # of orbit(s) and min., max. occupation numbers for restriction
```

```
# of orbit(s) for restriction?  (<CR> to quit):  2,5
```
陽子 d5/2 軌道と中性子 d5/2 軌道を指定。

```
  min., max.  restricted occupation numbers for the orbit(s) (or max only)
:  6,8
```
陽子 d5/2 軌道と中性子 d5/2 軌道の占有粒子数の合計の最小・最大値を指定。

`# of orbit(s) for restriction? (<CR> to quit): _` 続けて複数の制限を加えられます。

1. 各一粒子軌道の占有数の上限・下限によって制限します。
2. hw truncation は、調和振動子の励起数 ($N\hbar\omega$) によって制限します。
3. は、1. と 2. の両方を同時に指定して制限します。

独自にパーティションファイルを作成すれば、任意のトランケーションを実現できます。

4.3 ジョブスクリプトの実行

生成したスクリプト (先の例では Mg24_w.sh) を実行すると、計算が始まります。(スクリプト名は kshell_ui.py の最後に、"Finish. Execute ./Mg24_w.sh" と表示されます。)

MPI 並列の場合、ジョブ変数の指定や mpiexec の環境による動作の違いがありますので、hoge.sh を編集してください。それを用いてジョブを投入してください。(例えば、qsub hoge.sh)

計算終了後、summary_hoge.txt にエネルギー、遷移確率の計算結果の要約が保存されます。Q-moment などのより詳細な情報は、summary_hoge.txt に log-file として記されているファイル log.hoge を確認してください。E1 遷移,GT 遷移, 一粒子分光学的因子は、summary に収集されません。各 log ファイルの中身を参照してください。

5 詳細機能

5.1 相互作用ファイル

模型空間・有効相互作用は独自の .snt 形式に格納されています。kshell/snt に、いくつかの相互作用が例として含まれていますし、自作することも可能です。.snt 形式は、Nushell の .sp (模型空間定義ファイル) と.int(相互作用ファイル) を一つにまとめたような形式です。自作した場合、カレントディレクトリまたは kshell/snt に置いておけば認識します。

また、nushell2snt.py を用いることにより、Nushell(or OXBASH) 用の模型空間ファイルと相互作用ファイルを.snt 形式に変換出来ます。たとえば、

```
kshell/bin/nushell2snt.py sd.sp usda.int
```

とすると、フォーマットが変換された usda.snt が作成されます。[1]

5.2 相互作用ファイル .snt の詳細

一粒子軌道の定義、1 体相互作用行列要素、2 体相互作用行列要素と続きます。"!"から始まる行はコメントとみなされます。ハミルトニアンはエルミート性、時間反転対称性を仮定し、実数で与えられます。行列要素は以下の対称性をもち、反対称化されたもので、

[1] Nushell では、「int ファイル内では指定されていないが、実行する際には質量数依存性が導入される」というファイルがいくつかあります。このような int ファイルは、nushell2snt.py では対応できません。簡単な系を計算して、Nushell と KSHELL が同じ答えを出すかを確かめておくことをおすすめします。

重複を許さないように列挙します。アイソスピン対称性は仮定しません。アイソスピン対称性をもつ相互作用も、持たないことを仮定して記述します。

$$H = \sum_{\alpha,\beta,m_\alpha,m_\beta} \epsilon_{\alpha\beta} c^\dagger_{\alpha m_\alpha} c_{\beta m_\beta} + \sum_{\alpha\le\beta,\gamma\le\delta,J,M} \langle\alpha\beta|V|\gamma\delta\rangle_J A^\dagger(\alpha\beta JM) A(\gamma\delta JM) \quad (1)$$

$$A^\dagger(\alpha\beta JM) = \frac{1}{\sqrt{1+\delta_{\alpha\beta}}} \sum_{m_\alpha,m_\beta} \langle j_\alpha, m_\alpha, j_\beta . m_\beta | JM\rangle c^\dagger_{j_\alpha,m_\alpha} c^\dagger_{j_\beta,m_\beta}$$

$$\epsilon_{\alpha\beta} = \epsilon_{\beta\alpha} \quad (2)$$

$$\langle\alpha\beta|V|\gamma\delta\rangle_J = \langle\gamma\delta|V|\alpha\beta\rangle_J \quad (3)$$

$$\langle\alpha\beta|V|\gamma\delta\rangle_J = -(-1)^{j_\gamma+j_\delta-J}\langle\alpha\beta|V|\delta\gamma\rangle_J \quad (4)$$

例：w.snt

```
! "!"から始まる行はコメント行です。
!
! model space
! 陽子軌道数 中性子軌道数 閉殻陽子数 閉殻中性子数
   3   3     8   8
! 軌道番号 n   l   j  tz
   1       0   2   3  -1    !  1 = p 0d_3/2
   2       0   2   5  -1    !  2 = p 0d_5/2
   3       1   0   1  -1    !  3 = p 1s_1/2
   4       0   2   3   1    !  4 = n 0d_3/2
   5       0   2   5   1    !  5 = n 0d_5/2
   6       1   0   1   1    !  6 = n 1s_1/2
! 1体相互作用
! 行数 method1
     6   0
!  i   j      <i|V|j>
   1   1      1.64658
   2   2     -3.94780
   3   3     -3.16354
   4   4      1.64658
   5   5     -3.94780
   6   6     -3.16354
! 2体相互作用
!  行数 method2  A   mass dependence factor
   158   1  18  -0.30000
!  i   j   k   l     J      <i,j| V | k,l>_J
   1   1   1   1     0     -2.18450
   1   1   1   1     2     -0.06650
```

... 158 行続く
TBME は <i,j| V | k,l>_J で指定し、アイソスピンは仮定しない。
TBME の質量依存性は、
  method2 = 0　質量依存性なし
  method2 = 1　(質量数/A)^(mass dependence factor)

5.3 “kshell” バイナリ用のパラメータ

慣れてくると、kshell_ui.py で生成されたシェルスクリプトを自前で編集したくなると思います。以下、実行バイナリ “kshell” の namelist によるインプット変数を説明します。括弧内の数字はデフォルト値です。

- mtot 系全体の Jz 量子数を 2 倍して指定。
- n_eigen (1) 求めるべき固有状態の数。
- n_restart_vec (10) Thick-restart Lanczos における最小ベクトル数。
- max_lanc_vec (100) Thick-restart Lanczos における最大ベクトル数。
- maxiter (300) Thick-restart Lanczos における最大の反復回数。ここでいう反復回数とは、Lanczos 反復で max_lanc_vec までベクトル数が到達した後、n_restart_vec に圧縮して再開する回数を指す。
- hw_type (1) 調和振動子の $\hbar\omega$ 値を決定するための経験的式。
  $1 \cdots \hbar\omega = 41A^{-1/3}$　$2 \cdots \hbar\omega = 45A^{-1/3} - 25A^{-2/3}$
- is_double_j (.true.) Lanczos 反復中に J 射影 (2J=mtot) を行うか否か。
- is_load_snapshot (.false.) 途中経過を保存しているスナップショットファイル (tmp_*) から計算を再開するか否か。
- fn_save_wave 出力波動関数ファイル名
- fn_load_wave (ランダムに生成) ランチョス法における初期ベクトルが格納されているファイル名。
- beta_cm (0.0) Lawson 法による重心期待値除去のための変数。無次元化されていない (beta_cm= $\beta_{CM}\hbar\omega/A$, Nushell/OXBASH と同じ定義)。

$$H = H_{\rm int} + \beta_{\rm CM}(H_{\rm CM} - \frac{3}{2}\hbar\omega) \tag{5}$$

- eff_charge (1.0, 0.0) E2 遷移、四重極能率の計算に用いられる有効電荷。
- gl (1.0, 0.0), gs (5.586, -3.826) M1 遷移、磁気能率の計算に用いられる軌道 (gl), スピン (gs)g 因子。

- tol (0.000001) 収束条件。ランチョス反復において 1 反復あたりのエネルギー期待値の変化が tol 以下になると収束したと判断して反復を終了する。
- mode_lv_hdd (0) 再直交化に必要なランチョスベクトルをメモリーに保存 (0) するか、ハードディスクに保存 (1) するかを指定。

5.4 パーティションファイル詳細

先の例のように、^{24}Mg 正パリティ、$0d_{5/2}$ から 2 粒子励起までを許すようなトランケーションにおけるパーティションファイル (Mg24_w_p.ptn) を以下に示す。

```
# # から始まるのはコメント行
# particle-hole trucation orbit(s) : min., max.
#      [2, 5] :  6 8
# partition file of w.snt  Z=4  N=4  parity=+1
 4 4 1   # 陽子数、 中性子数、パリティ
# num. of  proton partition, netron partition
 6 6     # 陽子、中性子の占有粒子数組み合わせ（以下に続く行数）
# proton partition
     1     0  2  2  # 番号、 陽子 d3/2, d5/2, s1/2 の占有粒子数
     2     0  3  1
     3     0  4  0
     4     1  2  1
     5     1  3  0
     6     2  2  0
# neutron partition
     1     0  2  2  # 番号、中性子 d3/2, d5/2, s1/2 の占有粒子数
     2     0  3  1
     3     0  4  0
     4     1  2  1
     5     1  3  0
     6     2  2  0
# partition of proton and neutron
15            # 陽子・中性子占有粒子数の組み合わせ（以下に続く行数）
    1     3   # 陽子 1(0,2,2) - 中性子 3 (0,4,0)
    2     2
    2     3
    2     5
    3     1
    3     2
    3     3
    3     4
    3     5
```

```
3      6
4      3
5      2
5      3
5      5
6      3
```

正パリティ状態と、負パリティ状態のパーティションファイルは常に別個に用意する必要があることに注意。

5.5 制限事項

1. 陽子あるいは中性子の模型空間中の一粒子状態数の上限が 62 (Intel Fortran compiler など integer(16) をサポートしていない Fortran compiler を用いる場合)。gfortran を用い、constant.f90 中の変数を kmbit = 16 と指定してコンパイルすれば、上限は 126。

2. J-射影なしで M=0 状態を計算すると、同じ J 状態間の M1/E1 遷移確率と、磁気モーメントを求めることができません。(表示されません。) これは、Clebsch-Gordan 係数の性質 $\langle J, 0, 1, 0|J, 0\rangle = 0$ によるものです。求めたければ、 J-射影による計算、あるいは、M=1 状態を計算してください。

6 FAQ

1. 「popcnt, poppar が見つからない」というエラーでコンパイルできない。

   Makefile 中、下記の行の先頭 “#” を消す。

   ```
   # FFLAGS := $(FFLAGS) -DNO_POPCNT  # avoid using popcnt, poppar
   ```

2. Segmentation Fault と表示されてプログラム実行が停止する。

   あり得る原因の一つはスタック領域が不足していることです。 `ulimit -s unlimited` とする、Intel Fortran compiler の場合、“-heap-arrays” オプションをつけてコンパイルする。あるいは、OpenMP スレッド用のスタック領域が不足している可能性があります。kshell_ui.py で生成されたシェルスクリプトにて

   ```
   export OMP_STACKSIZE=1g
   ```

   が指定されているかを確認する。指定されていたら外してみる。また、指定されている数字を減らす・増やしてみる事も効果があるかもしれません。

3. “ERROR: increase max_ptnn” などと表示されてプログラム実行が停止する。

   bridge_partitions.F90 内の

```
integer, parameter :: max_idx=20, max_ptnn=300000
```

とある行、max_ptnn を増やす。他の変数の場合も同様。

4. M-scheme / J-scheme 次元をカウントしたい。

   snt ファイルと、パーティションファイル (.ptn) を用意し、kshell/bin/count_dim hoge.snt hoge.ptn とすることで、次元のみをカウントすることもできます。

5. メモリーがあふれてしまう。

   仕様上、少なくともランチョスベクトルを 2 本メモリーに載せる必要があります。M-scheme 次元を D とすると、倍精度実数を用いた計算では $D \times 16$ Byte のメモリ容量が必要になります。例えば、pf-shell 56Ni では $D = 10^9$ 次元なので、少なくとも 16GB のメモリー容量が必要になります。M-scheme 次元は、log ファイル途中に

   ```
   total m-scheme dimension      1087455228  = 10** 9.04
   ```

   と表示されます。必要な 1 ノードあたりのランチョスベクトルのメモリ容量は、log ファイルに下記のように表示されます。

   ```
   Memory / process is:     4.051 GB x     10 =     40.511 GB
   ```

   ランチョスベクトル以外の情報のストアのため、余裕をもってこの数字のおよそ 2 割増しで推定するといいでしょう。

   (a) 計算途中のランチョスベクトルをハードディスクに保存する。
   kshell_ui.py にて、パラメータ mode_lv_hdd を 1 に指定。
   (single node ではデフォルト 1 になっています)

   (b) 波動関数を単精度実数で計算する。
   メモリー使用量がほぼ半分になります。constant.f90 中の下記の行で指定されている kwf を 下記のように kwf=4 に変更。

   ```
   !  integer, parameter :: kwf = 8
     integer, parameter :: kwf = 4
   ```

   単精度実数で計算する場合、小さな M-scheme 次元における J 射影計算において希に計算が止まってしまうという問題が確認されています。その場合、倍精度に戻してください。

6. MS-Windows/Mac 上で、“libgomp: Thread creation failed: Resource temporarily unavailable” と表示されて止まってしまう。

   スレッド並列を始めるとき、メモリー容量が不足しています。主に、スタック領域の確保に問題がおこります。実行シェルスクリプト中の export OMP_STACKSIZE=1g とあるのを削除、 あるいは数字を増減してみてください。Makefile にて、FFLAGS コンパイルオプション “-Wl,–stack,40000000” を追加してください。(コメントアウトさ

れています。)] 状況が変わらなければ数字を増やす、“insufficient memory” と表示されて止まる場合は数字を減らしてください。スレッド数を減らしてみることも効果があるかもしれません。(たとえば4スレッドにするには、export OMP_NUM_THREADS=4とします。デフォルトでは、ハイパースレッディングによって見かけ上、物理コア数より大きなスレッド数が使われることがあります。しかしながら、ハイパースレッディングは計算速度に寄与しませんので、物理コア数を指定すれば十分です。) それでも駄目なら OpenMP を無効にしてください (`export OMP_NUM_THREADS=1`)。複数のコアを有効活用するには、OpenMP 無効＋ openmpi で、flat MPI 並列計算をする方法もあります。

7 謝辞

原子力研究開発機構の宇都野穣氏、専修大の水崎高浩氏、会津大の本間道雄氏に本ソフトウェアの設計段階からの議論と有益なコメントに感謝します。特に宇都野氏には一部のコードを寄与いただきました。また、東大理の大塚孝治氏には有益なコメントと開発の際の計算機資源についてご協力いただきました。東大理の角田佑介氏、角田直文氏にはテストに協力いただきました。OXBASH/Nushell のユーザーインターフェース、MSHELL/MSHELL64 の設計は大変参考になりました。他、多数の方々とのこれまでの共同研究に感謝します。

また、このソフトウェアの開発には、科研費 17K05433, 25870168, 15K05094、理研ー東大 CNS 大規模核構造計算共同プロジェクト、HPCI 戦略プログラム分野 5「物質と宇宙の起源と構造」、ポスト京重点課題 9，富岳成果創出加速プログラムの援助を受けました。東京大学 FX10、筑波大学 CCS COMA、理研 AICS 京計算機 (hp130024, hp140210) を用いています。