Priority Issue 9
to be Tackled by Using Post K Computer
"Elucidation of the Fundamental Laws
and Evolution of the Universe"
KAKENHI grant 17K05433, 25870168

CNS Summer school 2019
2019/08/21-27, Hongo, The University of Tokyo

Nuclear shell model calculations – basics and practices –

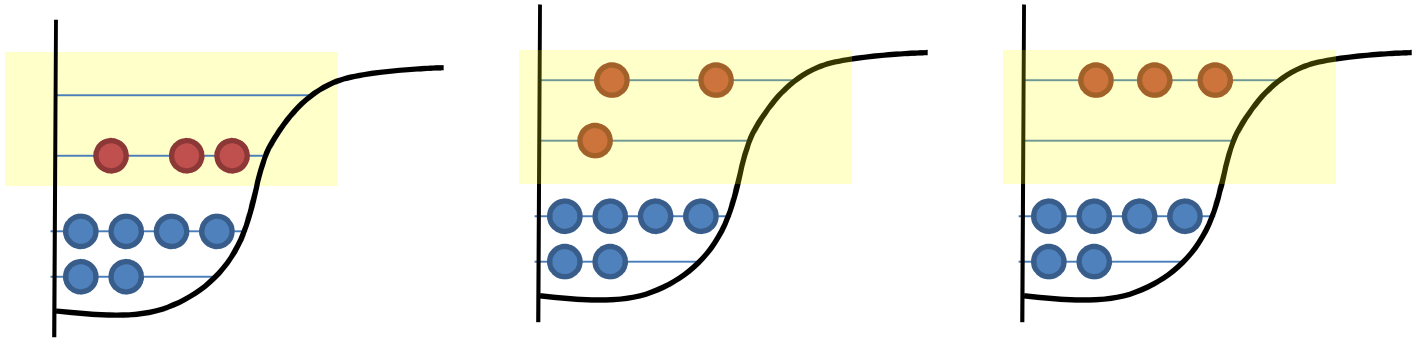
2. shell model code "KSHELL"

Noritaka Shimizu

Center for Nuclear Study,
the University of Tokyo

Large-scale shell model calculation (LSSM)

- Consider the inert core and active particles in the valence shells (model space
- Nuclear wave function is expressed as a linear combination of M-scheme basis states



$$|\Psi\rangle = v_1|m_1\rangle + v_2|m_2\rangle + v_3|m_3\rangle + \cdots$$

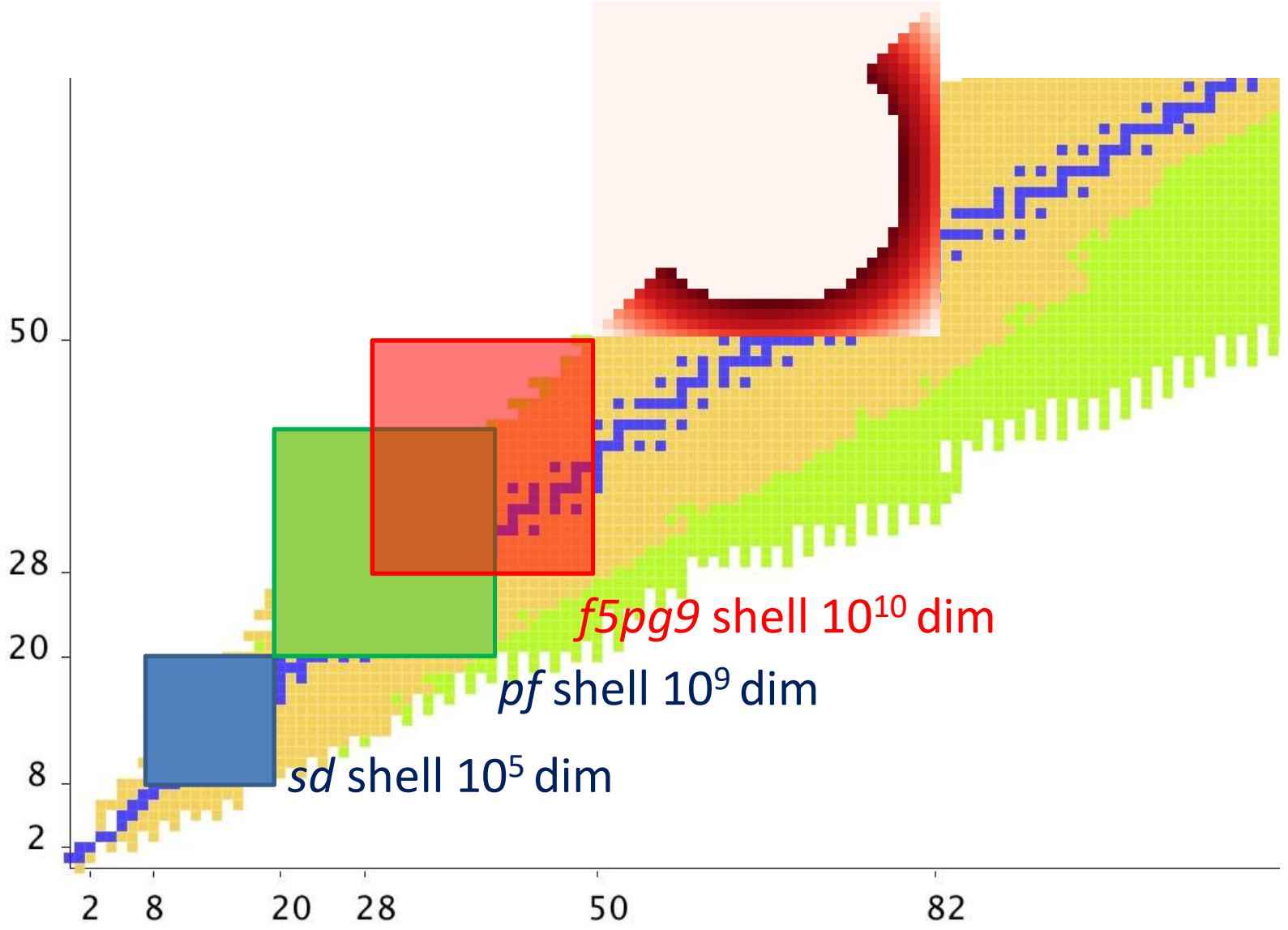
$$|\Psi\rangle = \sum_m v_m |m\rangle$$

Schrodinger's equation
... Eigenvalue problem of huge sparse matrix

$$\sum_k \langle m|H|k\rangle v_k = E v_m$$

Solved to obtain low-lying eigenstates

Applicability of shell-model calc.



Various shell-model codes

---------- single node ----------

- OXBASH/NuShell @MSU/Oxford
  - public, user interface, manual, OpenMP
  - *JT*-scheme
- ANTOINE / NATHAN @Strasbourg
  - public (ANTOINE only), highly tuned, single core
  - *M*-scheme / *J*-scheme
- MSHELL / MSHELL64 @Senshu T. Mizusaki et al.
  - *M*-scheme, unpublic
- Oslo code, CMichSM(CMU), EICODE(Jyvaskyla), jjSMQ(Kyusyu), ...

---------- MPI Parallel ----------

- BIGSTICK (San Diego), MFDn (Iowa, no core),....
  - supported by SciDAC UNEDF

KSHELL can be used in a simple way, M-scheme
From single PC to MPI+OpenMP supercomputer

shell-model code “KSHELL”

- *M*-scheme shell-model code
- Ref. N. Shimizu, T. Mizusaki, T. Utsuno, and Y. Tsunoda, Comp. Phys. Comm. in press. https://doi.org/10.1016/j.cpc.2019.06.011
- MPI + OpenMP hybrid parallel, also useful for a PC
- Thick-restart block Lanczos method

- Awkward in no-core shell model calc., 3-body force is out of focus

Benchmark

46Ti, pf-shell, KB3 interaction $D_M = 56{,}349$

Elapsed time to obtain J=4 10 lowest states

(J=4,T=4 10 lowest states for OXBASH)

@ Xeon E5-2680v2 2.80GHz, 20 CPU cores

- OXBASH 227.3 sec.
- KSHELL 117.5 sec. 1 thread, block size=1
- KSHELL 8.6 sec. 20 threads, block size=1
- KSHELL 3.5 sec. 20 threads, block size=6

KSHELL how to

https://sites.google.com/a/cns.s.u-tokyo.ac.jp/shimizu/cns-summer-school-2019

Ref. N. Shimizu *et al.,* Comp. Phys. Comm. in press.

https://doi.org/10.1016/j.cpc.2019.06.011

Computational Environment

- Software requirement
  - Fortran compiler
  - BLAS, LAPACK library
  - python 2
  - Linux OS, Mac OS + Xcode, Windows subsystem for Linux
  - optional : Matplotlib and X server for drawing E2 map

- Hardware requirement
  - Large memory
    - restricts the *M*-scheme dimension
    - two Lanczos vectors 10^9dim = 8GB
  - Many CPU cores
    - performance $\propto$ number of CPU cores
  - Large HDD
    - 100 Lanczos vectors 10^9dim = 400GB

How to install

tar xvzf kshell-cpc.tar.gz

cd kshell-cpc/src

make single

alias kshell_ui.py=(installed dir)/kshell-cpc/bin/kshell_ui.py

That is all, if you are lucky.

How to run

1. kshell_ui.py

   ... answer questions to generate a shell script

2. run the generated script

Example: ^{28}Si with USD interaction
  6 protons, 6 neutrons with ^{16}O core
  93,710 M-scheme dim.   8.2 seconds

Demonstration

- 28Si in sd-shell

Count dimension

- count M-scheme and J-scheme dimension

../bin/count_dim.exe w.snt Si28_w_p.ptn

```
 Z=  6 N=  6 parity  1


      2*M        M-scheme dim.          J-scheme dim.
dim.    28                    1                     1   1.00x10^ 0  1.00x10^ 0
dim.    26                   18                    17   1.80x10^ 1  1.70x10^ 1
dim.    24                  123                   105   1.23x10^ 2  1.05x10^ 2
dim.    22                  472                   349   4.72x10^ 2  3.49x10^ 2
dim.    20                 1439                   967   1.44x10^ 3  9.67x10^ 2
dim.    18                 3560                  2121   3.56x10^ 3  2.12x10^ 3
dim.    16                 7619                  4059   7.62x10^ 3  4.06x10^ 3
dim.    14                14310                  6691   1.43x10^ 4  6.69x10^ 3
dim.    12                24210                  9900   2.42x10^ 4  9.90x10^ 3
dim.    10                37086                 12876   3.71x10^ 4  1.29x10^ 4
dim.     8                52175                 15089   5.22x10^ 4  1.51x10^ 4
dim.     6                67560                 15385   6.76x10^ 4  1.54x10^ 4
dim.     4                81122                 13562   8.11x10^ 4  1.36x10^ 4
dim.     2                90338                  9216   9.03x10^ 4  9.22x10^ 3
dim.     0                93710                  3372   9.37x10^ 4  3.37x10^ 3

Estimated memory size for single-node mode :         0.002GB
```

output : summary_Si28_w.txt

Energy relative to ^{16}O core

Excitation energy

Energy levels

	N	2J	prty	N_Jp	2T	E(MeV)	Ex(MeV)
0^+_1	1	0	+	1	0	-135.938	0.000
2^+_1	2	4	+	1	0	-133.950	1.987
4^+_1	3	8	+	1	0	-131.279	4.659
0^+_2	4	0	+	2	0	-130.927	5.011
3^+_1	5	6	+	1	0	-129.771	6.167
4^+_2	6	8	+	2	0	-128.901	7.037
0^+_3	7	0	+	3	0	-128.699	7.239
2^+_2	8	4	+	2	0	-128.415	7.522
2^+_3	9	4	+	3	0	-128.032	7.906
1^+_1	10	2	+	1	0	-127.998	7.940

B(E2) larger than 1.0 e^2 fm^4

2Ji	Ei	2Jf	Ef	Ex
0+(1)	-135.938	4+(1)	-133.950	1.987
0+(1)	-135.938	4+(3)	-128.032	7.906
8+(2)	-128.901	4+(2)	-128.415	0.485
8+(2)	-128.901	4+(3)	-128.032	0.869

Experiment (Nudat2)

E_{level} (keV)	Jπ
0.0	0+
1779.030 11	2+
4617.86 4	4+
4979.92 8	0+
6276.20 7	3+
6690.74 15	0+
6878.79 8	3-
6887.65 10	4+
7380.59 9	2+
7416.26 9	2+
7799.01 9	3+

output : log_Si28_w_m0p.txt

```
 total # of partitions                  1679  = 10** 3.23
 total m-scheme dimension              93710  = 10** 4.97
  max. # dim. / a partition                    1156
  max local dim. / proc, average                  93710                    93710

 Memory for one global Lanczos vector:      0.000 GB
 Memory / process is:      0.000 GB x      10 =       0.003 GB
 Total Memory for Lanczos vectors:      0.003 GB
 ...
 ...
 ...
 ------------------------------------------------------
    1  <H>:  -135.93772  <JJ>:      0.00000  J:  0/2  prty  1
                         <TT>:      0.00000  T:  0/2
  <p Nj>  0.673  4.623  0.704
  <n Nj>  0.673  4.623  0.704
 ------------------------------------------------------
    2  <H>:  -133.95030  <JJ>:      6.00000  J:  4/2  prty  1
                         <TT>:      0.00000  T:  0/2
  <p Nj>  0.771  4.252  0.977
  <n Nj>  0.771  4.252  0.977
    <Qp>    10.375   <Qn>    10.375   <eQ>     20.751
 ------------------------------------------------------
 ....
```

Energy

0^+_1

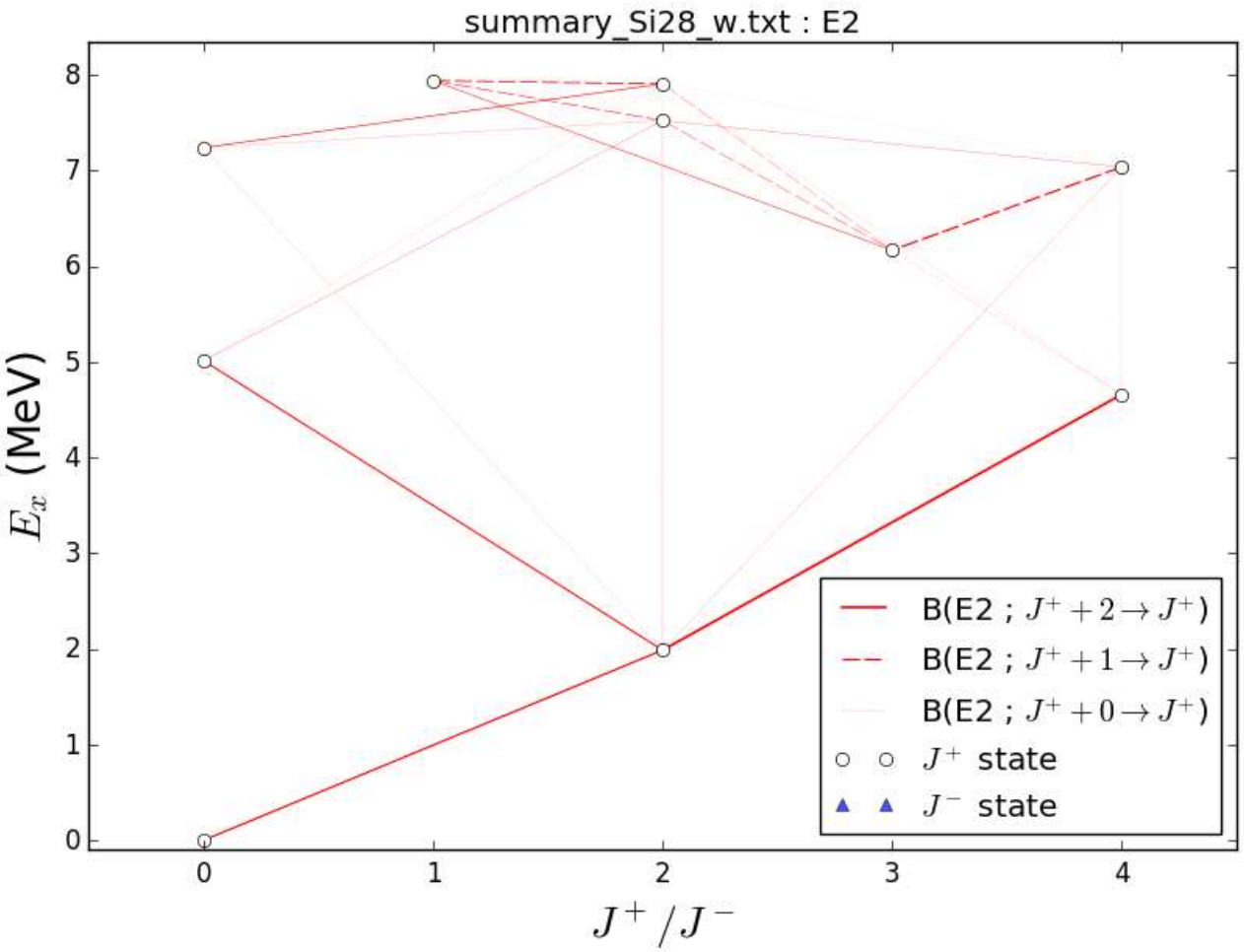
Occupation number of each orbit

2^+_1

Quadrupole moment

E2 map

./map_transit.py summary_Si28_w.txt

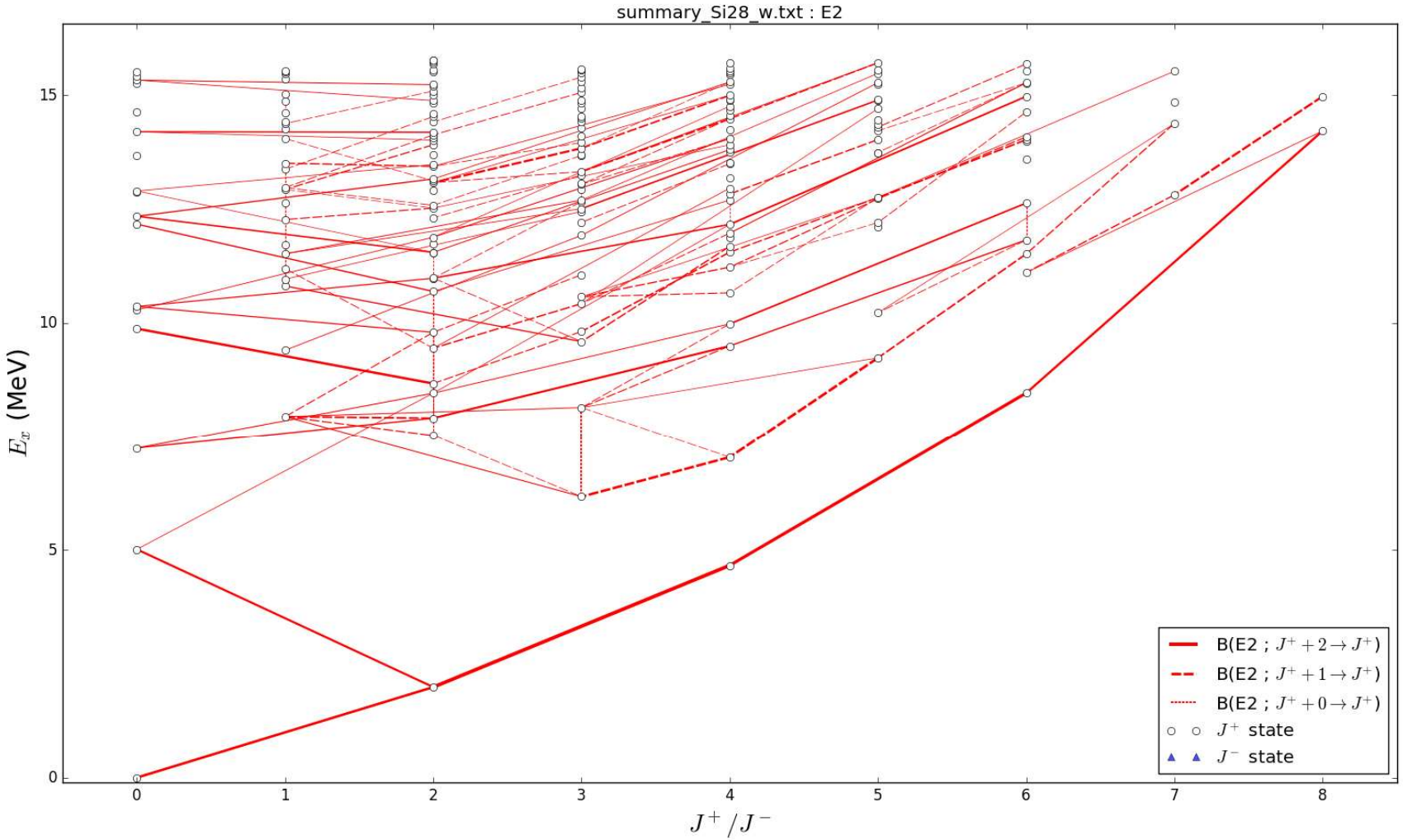


x-axis: J
y-axis : Ex. (MeV)

The width of the connected red line is proportional to B(E2) values.

This figure would help you to find hidden bands.

E2 map: ^{28}Si, 200 states



interaction and model space

kshell/snt

- w.snt ... Wildenthal's USD interaction (for sd shell)
- gxpf1a.snt ... GXPF1A interaction (for pf shell)
- jun45.snt ... JUN45 interaction (for f5pg9 shell Z,N=28-50)
- sdpf-mu.snt … SDPF-MU interaction (for sdpf shell)

GXPF1A, JUN45 Courtesy of Michio Honma (Aizu)
SDPF-M, SDPF-MU Courtesy of Yutaka Utsuno (JAEA)

"snt" file defines model space and its interaction.

You can make by your own snt file.

Some famous interaction files are equipped.

Interaction file in NuShell/OXBASH package can be transformed with "nushell2snt.py"

w.snt (USD interaction)

```
! Wildenthal's USD interaction for sd-shell
! B. A. Brown and B. H. Wildenthal, Annu. Rev. Nucl. Part. Sci. 38, 29 (1988
! proton-orbit, neutron-orbit, proton core, neutron core
3 3 8 8
! n  l j  tz
! model space
```

#	n,	l,	2j	2tz	
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

Model space
(*sd* shell)

```
! one-body interaction
! number of lines, method1
 6   0
```

1	1	1.64658
2	2	-3.94780
3	3	-3.16354
4	4	1.64658
5	5	-3.94780
6	6	-3.16354

Single-particle energy of each orbit
(one-body interaction)

w.snt (USD interaction) cont'd

Two-body matrix elements (TBME)

```
! two-body interaction (TBME)
! # of lines, 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
    1    1    1    2      2       0.61490
    1    1    1    3      2       0.51540
    1    1    2    2      0      -3.18560
    1    1    2    2      2      -1.62210
    1    1    2    3      2      -0.40410
    1    1    3    3      0      -1.08350
    1    2    1    2      1       1.03340
    1    2    1    2      2      -0.32480
    1    2    1    2      3       0.58940
    1    2    1    2      4      -1.44970
...
... 158 lines continued
```

$$\langle \pi 0d_{3/2}, \pi 0d_{3/2} | V | \pi 0d_{3/2}, \pi 0d_{3/2} \rangle_{J=0} = 2.1845\,(\mathrm{MeV})$$

Isospin symmetry is not assumed

```
--- 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 ) :
```

n_block = 4 … block size for block Lanczos method for acceleration

J-projection

- By default, KSHELL diagonalize the Hamiltonian in $M = 0$ subspace ($M = \frac{1}{2}$ for odd nuclei).
- It can also obtain only specified-*J* states by projecting the Lanczos vectors to good *J* states at every Lanczos iteration.

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


 number of valence protons and neutrons
  (ex.  2, 3 <CR> or 9Be <CR>)    <CR> to quit : Si28

 number of valence particles  6 6

 name for script file (default: Si28_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,
       1.5-1, 3.5+3 for lowest one 3/2- states and three 7/2+ states) :
```

Input parameters for shell-model calculations

- Model space and Hamiltonian
  - ask shell-model people!
- effective charges for *Q*-moment, B(E2)
  - $(e_\pi, e_\nu) = (1.5, 0.5)e$ is typical value, caused by the core polarization
- *g*-factor for *M*-moment, B(M1)
  - $g_{l\pi} = 1,\ \mathrm{g}_{l\nu} = 0,\ \mathrm{g}_{s\pi} = 5.586,\ \mathrm{g}_{s\nu} = -3.826$ for free particles
  - spin *g*-factor is typically quenched by 0.7, caused by the core polarization and meson exchange current
- $\hbar\omega$: Energy of the harmonic oscillator quanta
  - $\hbar\omega = 41\mathrm{A}^{-1/3}$
- Lawson's beta for removing contamination of center-of-mass motion (beyond $0\hbar\omega$ model space)
  - $\frac{\beta_{CM}\hbar\omega}{A} = 10.$ $\qquad H' = H + \beta_{\mathrm{CM}} H_{\mathrm{CM}}$

Try and have fun !

https://sites.google.com/a/cns.s.u-tokyo.ac.jp/shimizu/cns-summer-school-2019

Any questions and comments are welcome.
If you have trouble, ask me or Yusuke Tsunoda-san during the school.

CNS summer school 2019
GCOE workshop
PPNS
埼玉大集中講義
研究紹介
サイトマップ

For CNS Summer School 2019 attendees:

In my lectures, I will explain how to use the shell-model code "KSHELL" and will demonstrate some shell-model calculations using the code.
It is recommended to install the KSHELL code on your note PC before the summer school.

"KSHELL code" installation manual:

Setup for the installation :
KSHELL requires Fortran compiler, BLAS and LAPACK libraries, and Python 2.7.
For Linux users, install gfortran, BLAS and LAPACK libraries
For Windows users, install "Windows subsystem for Linux" and install Ubuntu Linux.
For Mac users, the installation manual is here (thanks to Sota Yoshida!).

For Ubuntu users, execute "apt-get install gfortran libblas-dev liblapack-dev".

Installation of the KSHELL:
Download KSHELL source file from here and execute the followings: