Emulator of shell-model calculations via eigenvector continuation & a surrogate model for IMSRG

"Eigenvector continuation and related techniques in nuclear structure and reaction theory"
30 May – 2 June @CEA Saclay

Utsunomiya University

Sota Yoshida

syoshida@cc.utsunomiya-u.ac.jp

Outline

- ◆ EC + valence shell model
- ◆ A Julia package for nuclear structure calculations

  ShellModel.jl ➡ NuclearToolkit.jl

- ◆ IMSRG-Net: a surrogate model for IMSRG
- ◆ Summary

schematic of EC + valence shell model

Constructing approximate shell-model wavefunctions by eigenvector continuation

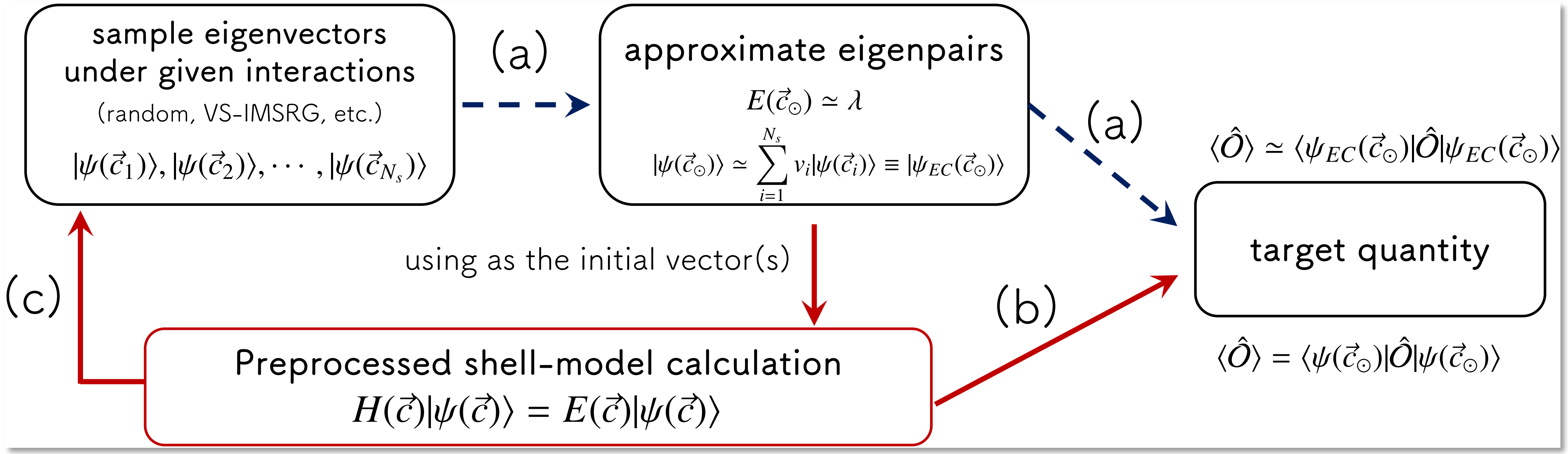

sample eigenvectors under given interactions
(random, VS-IMSRG, etc.)
$|\psi(\vec{c}_1)\rangle, |\psi(\vec{c}_2)\rangle, \cdots, |\psi(\vec{c}_{N_s})\rangle$
(a)
approximate eigenpairs
$E(\vec{c}_\odot) \simeq \lambda$
$|\psi(\vec{c}_\odot)\rangle \simeq \sum_{i=1}^{N_s} v_i |\psi(\vec{c}_i)\rangle \equiv |\psi_{EC}(\vec{c}_\odot)\rangle$
(a)
$\langle \hat{O} \rangle \simeq \langle \psi_{EC}(\vec{c}_\odot)|\hat{O}|\psi_{EC}(\vec{c}_\odot)\rangle$
target quantity
using as the initial vector(s)
(c)
(b)
$\langle \hat{O} \rangle = \langle \psi(\vec{c}_\odot)|\hat{O}|\psi(\vec{c}_\odot)\rangle$
Preprocessed shell-model calculation
$H(\vec{c})|\psi(\vec{c})\rangle = E(\vec{c})|\psi(\vec{c})\rangle$

FIG. 1

SY and N.Shimizu, PTEP 2022 053D02 (2022).

Problems of interest

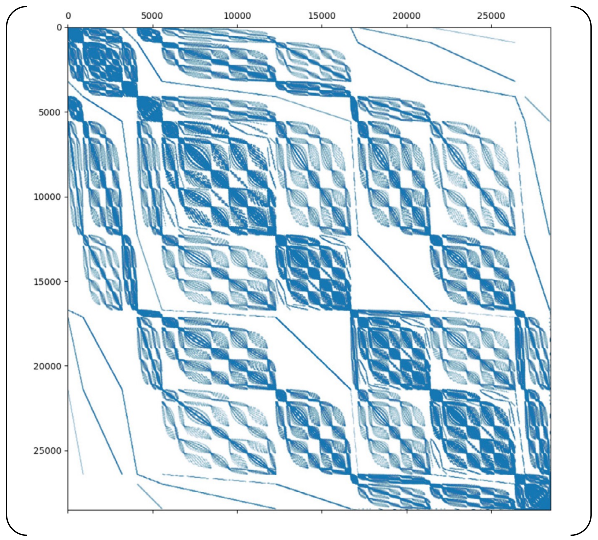
original problem:

$$H(\vec{c})|\psi(\vec{c})\rangle = E(\vec{c})|\psi(\vec{c})\rangle$$

$$H = H^{(1)} + H^{(2)} = \sum_{ac} \underset{\text{SPEs}}{\boxed{h^{(1)}_{ac}}} c^\dagger_a c_c + \frac{1}{4}\sum_{abcd} \underset{\text{TBMEs}}{\boxed{h^{(2)}_{abcd}}} c^\dagger_a c^\dagger_b c_d c_c,$$

large sparse matrix

e.g., ^{24}Mg M-scheme Dim. = 28,503



row & column:
many-body configurations

N. Shimizu et al., Comp. Phys. Comm. 244 (2019) 372–384

~ sec.

EC + shell model:

$$\tilde{H}\vec{v} = \lambda N \vec{v},$$
$$\tilde{H}_{i,j} = \langle \psi(\vec{c}_i)|H(\vec{c}_\odot)|\psi(\vec{c}_j)\rangle,$$
$$N_{i,j} = \langle \psi(\vec{c}_i)|\psi(\vec{c}_j)\rangle.$$

small dense matrix

Dim. = # of sample ~ 100

row & column:
indices for EC samples

~ msec. (or less?)

Sample eigenvectors

SY and N.Shimizu, PTEP 2022 053D02.

$$\tilde{H}\vec{v} = \lambda N \vec{v},$$
$$\tilde{H}_{i,j} = \langle \psi(\vec{c}_i) | H(\vec{c}_\odot) | \psi(\vec{c}_j) \rangle,$$
$$N_{i,j} = \langle \psi(\vec{c}_i) | \psi(\vec{c}_j) \rangle.$$

$$E(\vec{c}_\odot) \simeq \lambda,$$
$$|\psi(\vec{c}_\odot)\rangle \simeq \sum_{i=1}^{N_s} v_i |\psi(\vec{c}_i)\rangle \equiv |\psi_{EC}(\vec{c}_\odot)\rangle.$$

Example:

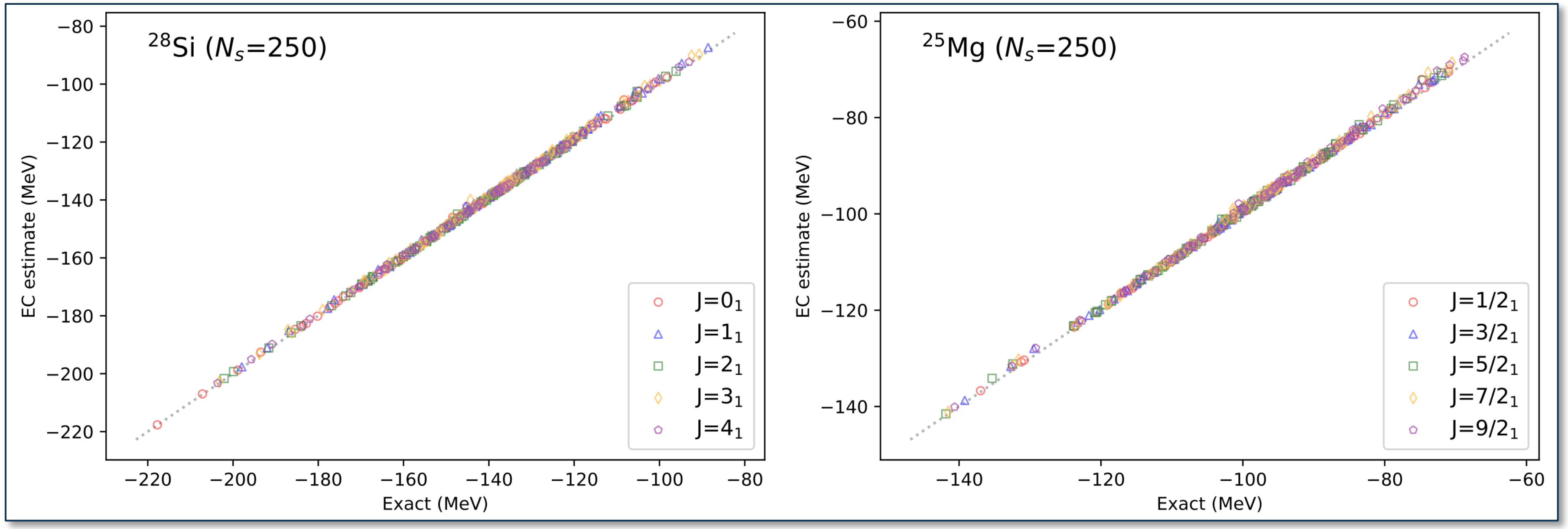
sd-shell (^{16}O core + 0d5/0d3/1s1 valence orbits)

parameters: 66 (3 SPEs & 63 TBMEs, w/ isospin)

target nuclei: ^{25}Mg (vp=4,vn=5), ^{28}Si (vp=vn=6, dim. ~ 90,000)

sampling 5 states for given total J at 50 (random) different points (5×50=250 samples) **around USDB**

"validation" for 100 random parameters



EC approximates energies within a few percent accuracy

Results: energy

SY and N.Shimizu, PTEP 2022 053D02.

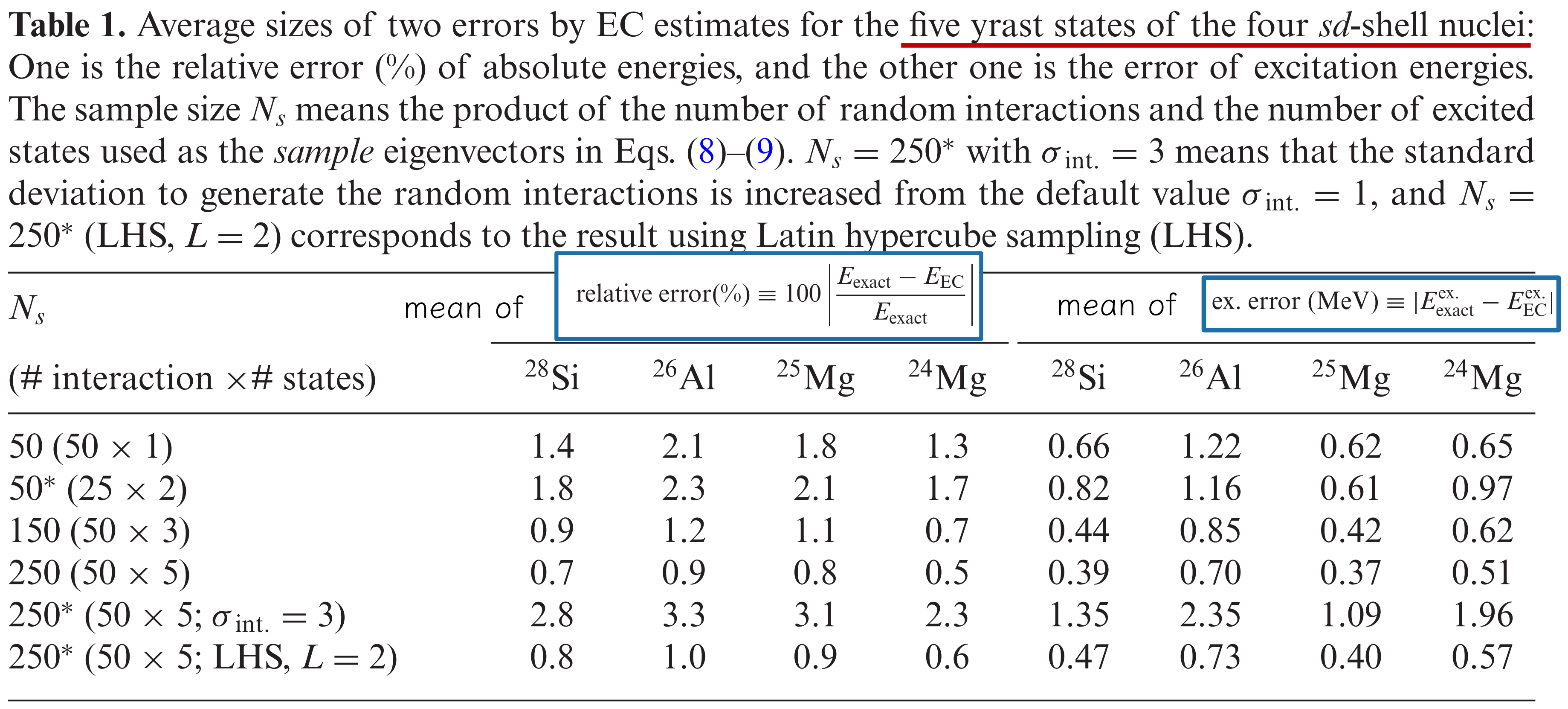


Table 1. Average sizes of two errors by EC estimates for the five yrast states of the four *sd*-shell nuclei: One is the relative error (%) of absolute energies, and the other one is the error of excitation energies. The sample size N_s means the product of the number of random interactions and the number of excited states used as the *sample* eigenvectors in Eqs. (8)–(9). $N_s = 250^*$ with $\sigma_{\text{int.}} = 3$ means that the standard deviation to generate the random interactions is increased from the default value $\sigma_{\text{int.}} = 1$, and $N_s = 250^*$ (LHS, $L = 2$) corresponds to the result using Latin hypercube sampling (LHS).

N_s	mean of $\text{relative error}(\%) \equiv 100 \left\lvert \frac{E_{\text{exact}} - E_{\text{EC}}}{E_{\text{exact}}} \right\rvert$				mean of $\text{ex. error (MeV)} \equiv \lvert E^{\text{ex.}}_{\text{exact}} - E^{\text{ex.}}_{\text{EC}} \rvert$			
(# interaction $\times$# states)	^{28}Si	^{26}Al	^{25}Mg	^{24}Mg	^{28}Si	^{26}Al	^{25}Mg	^{24}Mg
50 (50 $\times$ 1)	1.4	2.1	1.8	1.3	0.66	1.22	0.62	0.65
50* (25 $\times$ 2)	1.8	2.3	2.1	1.7	0.82	1.16	0.61	0.97
150 (50 $\times$ 3)	0.9	1.2	1.1	0.7	0.44	0.85	0.42	0.62
250 (50 $\times$ 5)	0.7	0.9	0.8	0.5	0.39	0.70	0.37	0.51
250* (50 $\times$ 5; $\sigma_{\text{int.}} = 3$)	2.8	3.3	3.1	2.3	1.35	2.35	1.09	1.96
250* (50 $\times$ 5; LHS, $L = 2$)	0.8	1.0	0.9	0.6	0.47	0.73	0.40	0.57

each row shows different settings (# of samples, way of sampling)

- sample not only g.s. but also excited states if you want to know excited states too
- odd or odd-odd nuclei are more difficult than even (even-even) ones

Results: lower pf-shell case

SY and N.Shimizu, PTEP 2022 053D02.

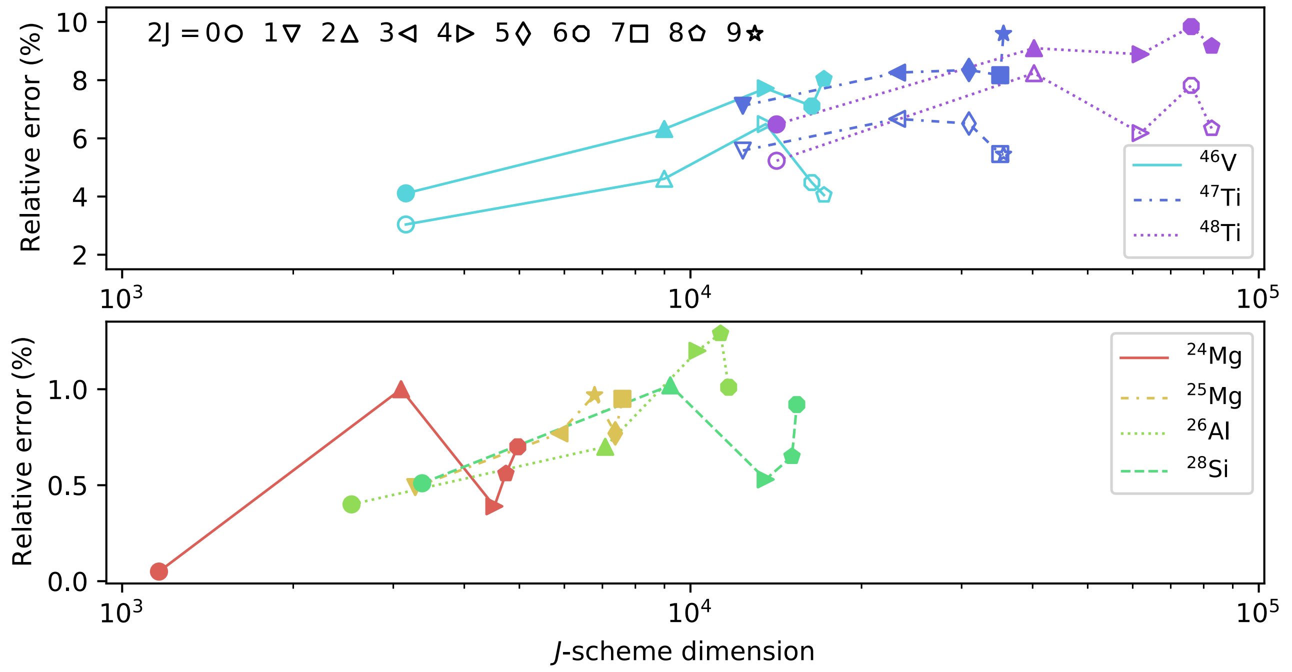


Fig. 7. Relative errors with $N_s = 250$ samples against J-scheme dimensions for ^{24,25}Mg, ^{26}Al, ^{28}Si (lower panel), ^{46}V, and ^{47,48}Ti (upper panel). The filled symbols correspond to the samples generated by varying all the parameters with $\sigma_{\text{int.}} = 1$ around the reference values (USDB and GXPF1A). The open symbols in the upper panel show the results with samples in which only the 32 parameters related to $f7/2$ and $p3/2$ were varied with the same $\sigma_{\text{int.}}$.

pf-shell => <u>**199 parameters**</u> (4 SPEs & 195 TBMEs)

(craziest application of EC method !?)

Dotted lines are results considering

only the samples spread over f7/2&p3/2

it is better to sample over a subspace

more relevant to what you want to know

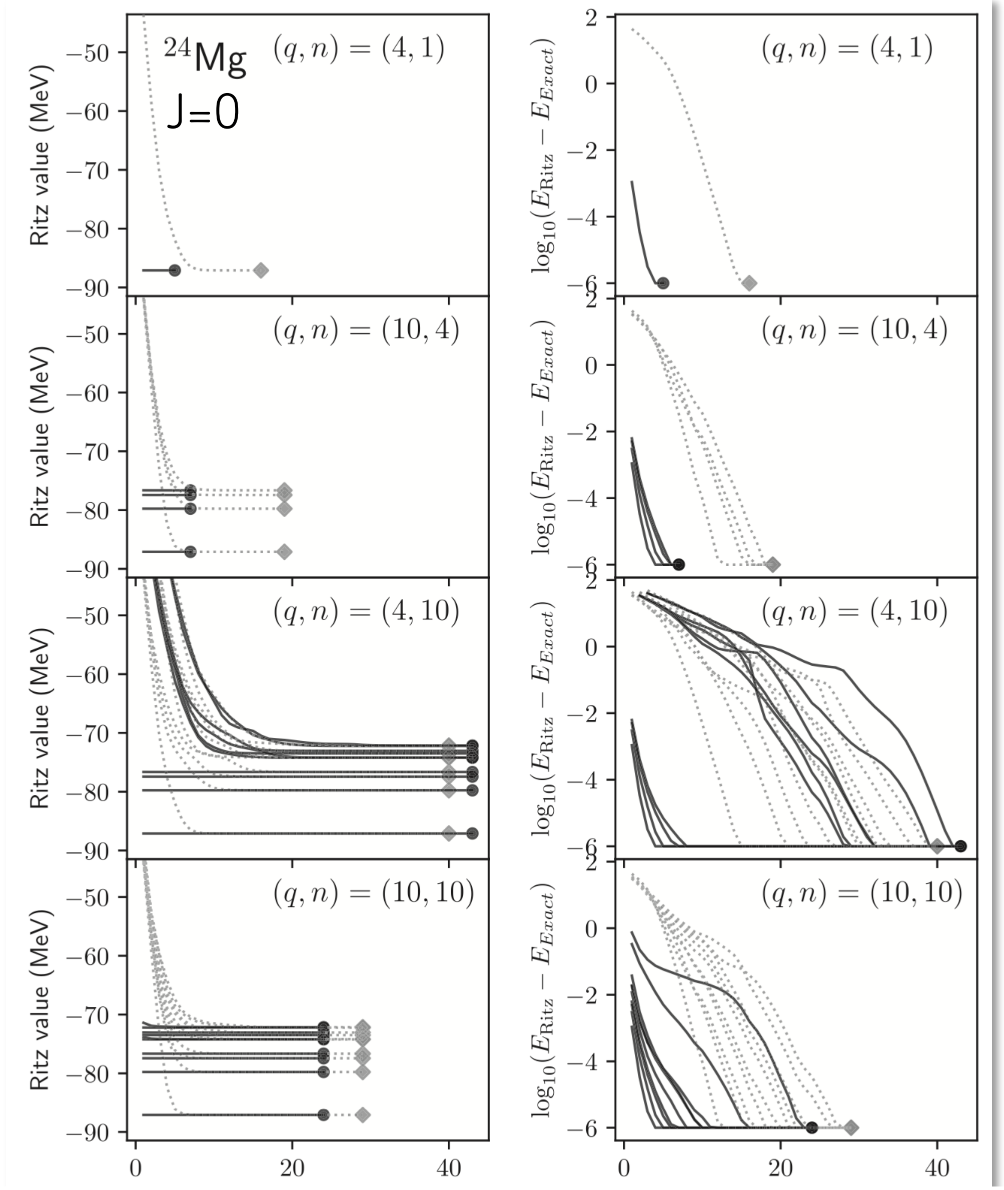
Q. What is a better sampling strategy?

Is there any way to maximize **information gain** from a next observation?

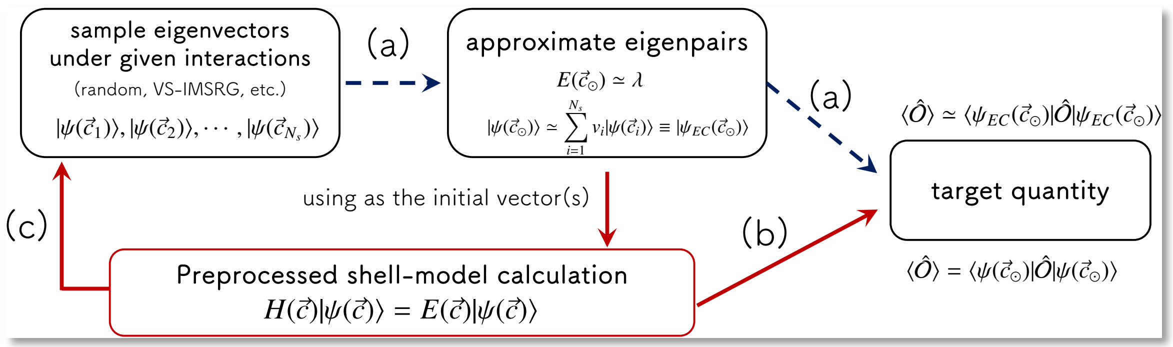
no answer will be shown in this talk though...

As a preprocessor

SY and N.Shimizu, PTEP 2022 053D02.



Number of H operation during the (block) Lanczos method



sample eigenvectors under given interactions
(random, VS-IMSRG, etc.)
$|\psi(\vec{c}_1)\rangle, |\psi(\vec{c}_2)\rangle, \cdots, |\psi(\vec{c}_{N_s})\rangle$
(a)
approximate eigenpairs
$E(\vec{c}_\odot) \simeq \lambda$
$|\psi(\vec{c}_\odot)\rangle \simeq \sum_{i=1}^{N_s} v_i|\psi(\vec{c}_i)\rangle \equiv |\psi_{EC}(\vec{c}_\odot)\rangle$
(a)
$\langle\hat{O}\rangle \simeq \langle\psi_{EC}(\vec{c}_\odot)|\hat{O}|\psi_{EC}(\vec{c}_\odot)\rangle$
target quantity
$\langle\hat{O}\rangle = \langle\psi(\vec{c}_\odot)|\hat{O}|\psi(\vec{c}_\odot)\rangle$
using as the initial vector(s)
(c)
(b)
Preprocessed shell-model calculation
$H(\vec{c})|\psi(\vec{c})\rangle = E(\vec{c})|\psi(\vec{c})\rangle$

q: size of initial "block" vector

n: # of excited states of interest

dotted: initialized by random vectors

solid: initialized by EC eigenvectors

Starting from better initial guess,
# of manipulation could be reduced!!

Exception => (q, n) = (4,10)

since the emulator is trained with 5 lowest states,
such emulator do not have much info. on higher states

converged results are obtained at ●/◆

To feed more samples…

SY and N.Shimizu, PTEP 2022 053D02.

Sampling itself is not easy …

$$\tilde{H}\vec{v} = \lambda N\vec{v},$$

$$\tilde{H}_{i,j} = \langle\psi(\vec{c}_i)|H(\vec{c}_\odot)|\psi(\vec{c}_j)\rangle, \quad \leftarrow \text{most time-consuming part}$$

$$N_{i,j} = \langle\psi(\vec{c}_i)|\psi(\vec{c}_j)\rangle.$$

- You don't need to explicitly calculate H(C$_\odot$)I ψ (cj) > for each parameter C$_\odot$ to evaluate H-tilde above:

all you need is 1&2-body transition densities

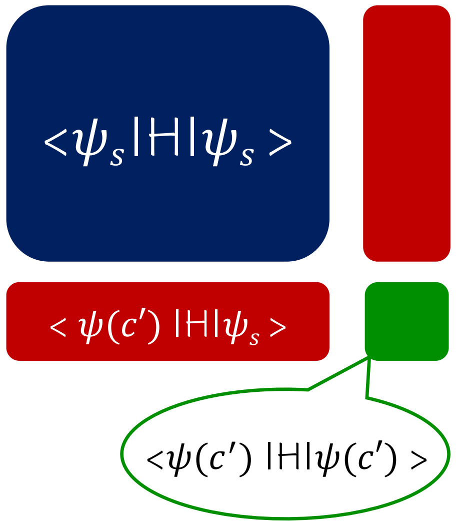
$$\tilde{H}_{i,j} = \sum_k \underbrace{\boxed{h^{(1)}_k}}_{\text{SPEs}} \times \overline{\text{OBTD}}_k{}^{\text{i,j}} + \sum_k \underbrace{\boxed{V_J(abcd)_k}}_{\text{TBMEs}} \times \overline{\text{TBTD}}_k{}^{\text{i,j}},$$

- If you want to increase sample number (for better accuracy),

  prepare new sample (green) and calc. overlap (transition densities)

  between new w.f. and previous samples (red)

s: sampled w.f.s



Outline

- ◆ EC + valence shell model
- ◆ A Julia package for nuclear structure calculations

  ShellModel.jl ➡ NuclearToolkit.jl

- ◆ IMSRG-Net: a surrogate model for IMSRG
- ◆ Summary

Why I developed NuclearToolkit.jl

※These are my personal opinions

- A single method (code) can be lengthy ~ 100,000 lines

  two language problem (Fortran/C++ & shell/Python)

- Especially in Japan, research methods are "clusterized" (localized)

  to a specific group. This can be an obstacle to

  collective intelligence or co-creation

  seen in e.g. ML community through ML frameworks

- Educations for next generation

  julia can circumvent the situation...?

secret sauce (source)
in Prof. XX Group

NuclearToolkit.jl: Julia package for structure calculations

SY, Journal of Open Source Software, 7(79), 4694,(2022)

- ChiEFTint ～ 8000 lines
  - ◆ NN potential, Entem-Machleidt(N3LO), EMN(EKMN, N4LO)
  - ◆ SRG in momentum space (NN-only)
  - ◆ effective NN from 3NF
  - ◆ valence NN interaction ≠ effective interaction
  - ◆ input for No-core shell model (in KSHELL fmt)
  - ◆ genuine 3NF (only in Jacobi HO form)

    ※plz use NuHamil (by Takayuki Miyagi@TUDarmstadt)

- HartreeFock ～ 3000 lines
  - ◆ spherical HF (from snt/snt.bin/memory)
  - ◆ HFMBPT Energy=> 3rd order, Scaler operator => 2nd order
  - ◆ Normal ordering w.r.t. target reference, ensemble normal ordering

- IM-SRG ~2700 lines

  free space
  - ◆ IMSRG(2) calculation => g.s. properties
  - ◆ consistent IMSRG flow of operators with Magnus expansion

  valence space (VS-IMSRG)
  - ◆ derive effective interaction for a target model space
  - ◆ effective operators (only scaler ones for now)

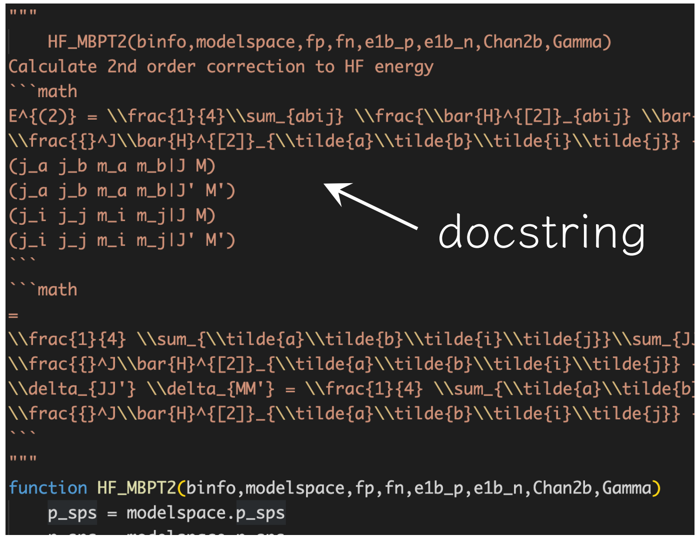
- ShellModel.jl ~5000 lines
  - ◆ eigenvector continuation (fast emulator of exact wavefunctions)

~ 20,000 lines
(including document)

Do you think it's lengthy?🤔

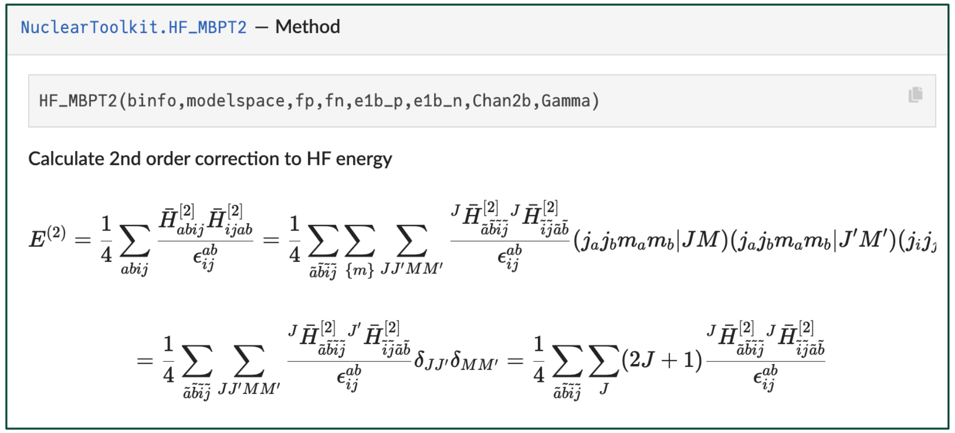
Documenter.jl

Docs are automatically generated from docstring (in markdown)



~~~
"""
    HF_MBPT2(binfo,modelspace,fp,fn,e1b_p,e1b_n,Chan2b,Gamma)
Calculate 2nd order correction to HF energy
```math
E^{(2)} = \\frac{1}{4}\\sum_{abij} \\frac{\\bar{H}^{[2]}_{abij} \\bar
\\frac{{}^J\\bar{H}^{[2]}_{\\tilde{a}\\tilde{b}\\tilde{i}\\tilde{j}}
(j_a j_b m_a m_b|J M)
(j_a j_b m_a m_b|J' M')
(j_i j_j m_i m_j|J M)
(j_i j_j m_i m_j|J' M')
~~~
```math
=
\\frac{1}{4} \\sum_{\\tilde{a}\\tilde{b}\\tilde{i}\\tilde{j}}\\sum_{J
\\frac{{}^J\\bar{H}^{[2]}_{\\tilde{a}\\tilde{b}\\tilde{i}\\tilde{j}}
\\delta_{JJ'} \\delta_{MM'} = \\frac{1}{4} \\sum_{\\tilde{a}\\tilde{b
\\frac{{}^J\\bar{H}^{[2]}_{\\tilde{a}\\tilde{b}\\tilde{i}\\tilde{j}}
```
"""
function HF_MBPT2(binfo,modelspace,fp,fn,e1b_p,e1b_n,Chan2b,Gamma)
    p_sps = modelspace.p_sps
```

docstring



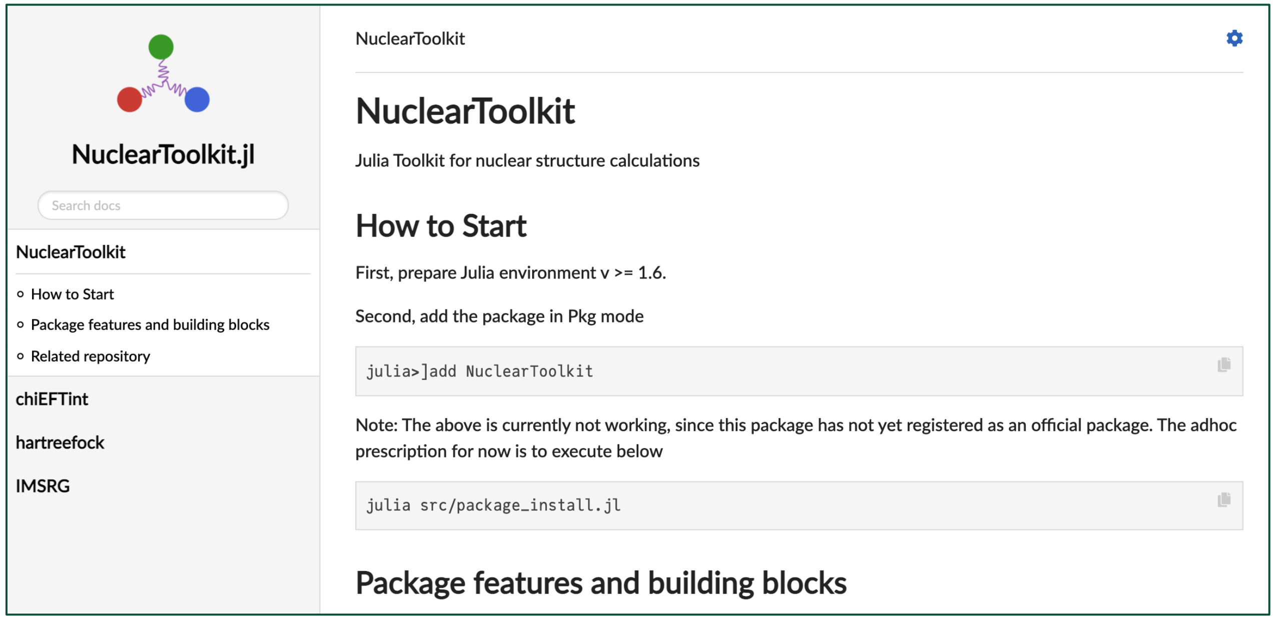
`NuclearToolkit.HF_MBPT2` — Method

```
HF_MBPT2(binfo,modelspace,fp,fn,e1b_p,e1b_n,Chan2b,Gamma)
```

Calculate 2nd order correction to HF energy

$$E^{(2)} = \frac{1}{4}\sum_{abij}\frac{\bar{H}^{[2]}_{abij}\bar{H}^{[2]}_{ijab}}{\epsilon^{ab}_{ij}} = \frac{1}{4}\sum_{\tilde{a}\tilde{b}\tilde{i}\tilde{j}}\sum_{\{m\}}\sum_{JJ'MM'}\frac{{}^J\bar{H}^{[2]}_{\tilde{a}\tilde{b}\tilde{i}\tilde{j}}\,{}^J\bar{H}^{[2]}_{\tilde{i}\tilde{j}\tilde{a}\tilde{b}}}{\epsilon^{ab}_{ij}}(j_aj_bm_am_b|JM)(j_aj_bm_am_b|J'M')(j_ij_j$$

$$= \frac{1}{4}\sum_{\tilde{a}\tilde{b}\tilde{i}\tilde{j}}\sum_{JJ'MM'}\frac{{}^J\bar{H}^{[2]}_{\tilde{a}\tilde{b}\tilde{i}\tilde{j}}\,{}^{J'}\bar{H}^{[2]}_{\tilde{i}\tilde{j}\tilde{a}\tilde{b}}}{\epsilon^{ab}_{ij}}\delta_{JJ'}\delta_{MM'} = \frac{1}{4}\sum_{\tilde{a}\tilde{b}\tilde{i}\tilde{j}}\sum_{J}(2J+1)\frac{{}^J\bar{H}^{[2]}_{\tilde{a}\tilde{b}\tilde{i}\tilde{j}}\,{}^J\bar{H}^{[2]}_{\tilde{i}\tilde{j}\tilde{a}\tilde{b}}}{\epsilon^{ab}_{ij}}$$



NuclearToolkit.jl

Search docs

- NuclearToolkit
  - How to Start
  - Package features and building blocks
  - Related repository
- chiEFTint
- hartreefock
- IMSRG

NuclearToolkit

NuclearToolkit

Julia Toolkit for nuclear structure calculations

How to Start

First, prepare Julia environment v >= 1.6.

Second, add the package in Pkg mode

```
julia>]add NuclearToolkit
```

Note: The above is currently not working, since this package has not yet registered as an official package. The adhoc prescription for now is to execute below

```
julia src/package_install.jl
```

Package features and building blocks
```

# Workflow w/ GitHub Actions

developer/user make
changes to code/docstring

pull request to GitHub repository

GitHub Actions

- Automatic generation and deployment of the Docs
- Execute test codes with specified OS / version of Julia

*avoiding destructive changes/releases*

- see the docs
- play with the code

```
using NuclearToolkit
your_own_function()
```

c.f. CI/CD: Continuous Integration／Continuous Delivery

How to start NuclearToolkit.jl (within 5 min.)

◆Installation of Julia

Download Julia binary and add to PATH

◆Installation of NuclearToolkit.jl

Download src (recommended)

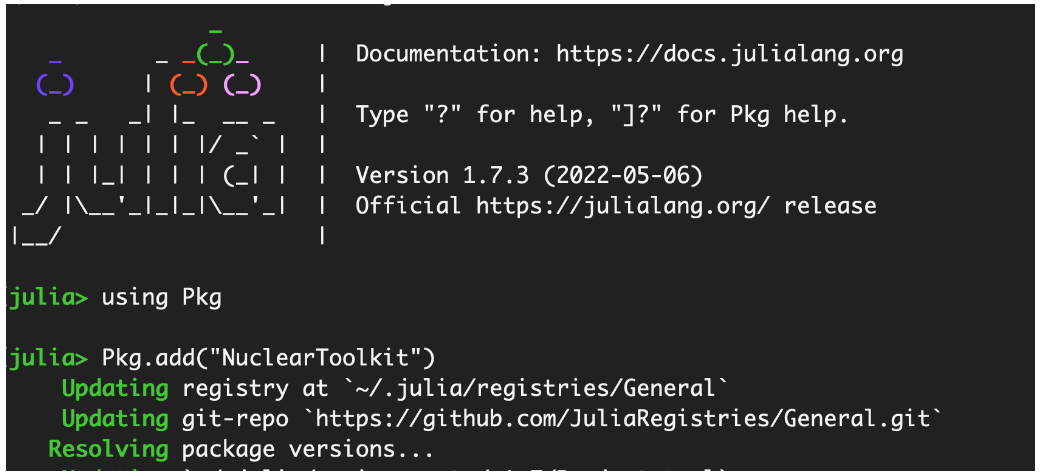
$git clone https://github.com/SotaYoshida/NuclearToolkit.jl

Note:

NuclearToolkit.jl was registered as a Julia package.

You can install the package in Julia's REPL

like "pip" in Python



```
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.7.3 (2022-05-06)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia> using Pkg

julia> Pkg.add("NuclearToolkit")
    Updating registry at `~/.julia/registries/General`
    Updating git-repo `https://github.com/JuliaRegistries/General.git`
   Resolving package versions...
```

Outline

- ◆ EC + valence shell model
- ◆ A Julia package for nuclear structure calculations

ShellModel.jl ➡ NuclearToolkit.jl

- ◆ IMSRG-Net: a surrogate model for IMSRG
- ◆ Summary

In-medium Similarity Renormalization Group (IMSRG)

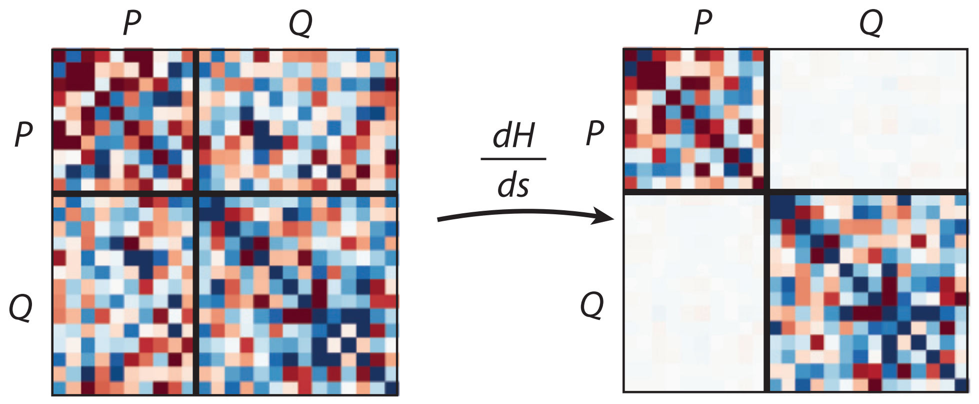
K. Tsukiyama, S. K. Bogner, and A. Schwenk, PRL **106**, 222502 (2011).
K. Tsukiyama, S. K. Bogner, and A. Schwenk, PRC **85**, 061304 (2012).

IMSRGflow:

$$\frac{dH(s)}{ds} = [\eta(s), H(s)],$$

$$H(s) \begin{cases} \text{0-body: scaler, } E_0(s) = < H(s) > \\ \text{1-body: (n,n) matrix (n = \# of sps, block structure)} \\ \text{2-body: } (\mathbf{d_i}, \mathbf{d_i})\text{matrix (i is label of \{J,P,Tz\})} \end{cases}$$

NO2B→IMSRG(2)



$H(s = 0, HF) \quad \Rightarrow \quad H(s)$

under HF reference state,

divide single particle states as

P: hole

Q: particle

S.R.Stroberg et al., Annu. Rev. Nucl. Part. Sci. 2019. 69:307–62 (2019)

"off-diagonal" component, P-Q $\quad \lim_{s\to\infty} H^{od}(s) = 0$

"diagonal" component, P-P, Q-Q $\quad \lim_{s\to\infty} H^{d}(s) = H_{\mathrm{eff}}. \quad \rightarrow$ better $E_0(s)$

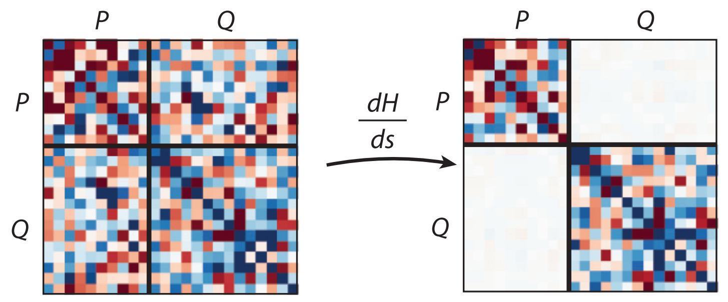
adopting a certain generator η to achieve this "decoupling"

Magnus formulation of IMSRG

T.D.Morris et al., PRC **92**, 034331 (2015)
S.R.Stroberg et al., Annu. Rev. Nucl. Part. Sci. 2019. 69:307–62 (2019)

$$H(s) = U^\dagger(s) H U(s) \quad \Leftrightarrow \quad \frac{dH(s)}{ds} = [\eta(s), H(s)],$$

$$O(s) = U^\dagger(s) O U(s) \quad \Leftrightarrow \quad \frac{dO(s)}{ds} = [\eta(s), O(s)],$$



formulation with Magnus expansion

$$U(s) = e^{\Omega(s)}$$

$$H(s+ds) = e^{\eta(s)ds} e^{\Omega(s)} H(0) e^{-\Omega(s)} e^{-\eta(s)ds}$$

$$e^{\Omega}(s+ds) \equiv e^{\eta(s)ds} e^{\Omega(s)}$$

$$\Omega(s+ds) = \Omega(s) + \eta(s)ds + \frac{1}{2}[\eta(s), \Omega(s)]ds + \frac{1}{12}[\Omega(s), [\Omega(s), \eta(s)]]ds + \ldots$$

explicit calculation of unitary transformation via Magnus operator Ω
- nested commutator (BCH formula)
- 1-step is enough (Euler method), nice for memory or I/O
- reuse U(s) = exp(Ω(s)) → Any operators can be evolved simultaneously

valence-space IMSRG (VS-IMSRG)

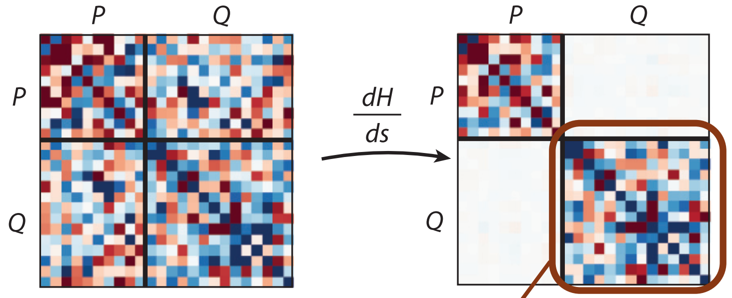
S.R.Stroberg,et al., PRL 118, 032502 (2017)
S.R.Stroberg et al., Annu. Rev. Nucl. Part. Sci. 2019. 69:307–62 (2019)

IMSRG: P: hole, Q: particle

"off-diagonal" component, P-Q $\lim_{s\to\infty} H^{od}(s) = 0$

"diagonal" component, P-P, Q-Q $\lim_{s\to\infty} H^{d}(s) = H_{\text{eff}}.$

adopting a certain generator η to achieve this "decoupling"

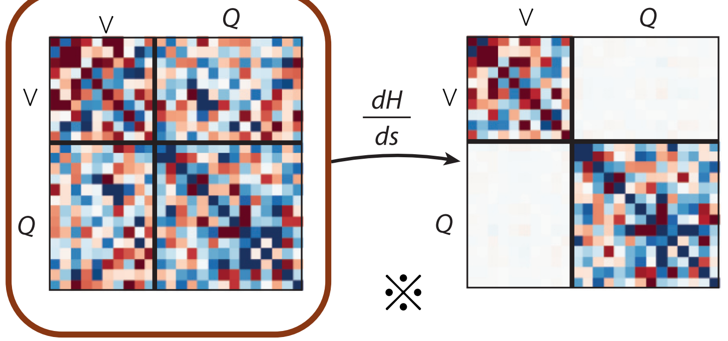


VS-IMSRG: P => valence, Q => q-space

$$\eta_{abij} = \frac{1}{2}\arctan\left(\frac{2\Gamma_{abij}}{f_{aa}+f_{bb}-f_{ii}-f_{jj}+G_{abij}+\Delta}\right),$$

$$G_{abij} = \Gamma_{abab}+\Gamma_{ijij}-(\Gamma_{aiai}+\Gamma_{bjbj}+[a\leftrightarrow b]).$$

denominator Delta: prescription for multi-shell interaction
see T. Miyagi et.al., PRC 102, 034320 (2020)



Hv(s→∞) = Effective interactions for a valence space

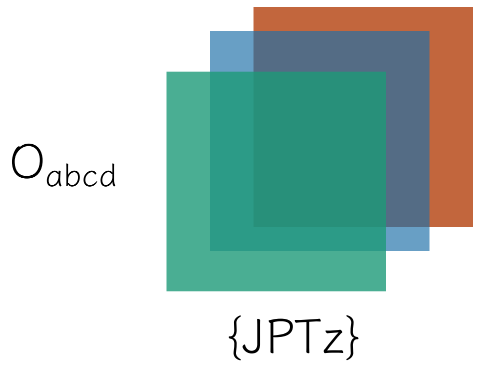
(※ core is still considered in VS-IMSRG calculations, this figure is edited (by me) to focus on v-space)

Surrogate models for IMSRG/VS-IMSRG…

One-body Hamiltonians <a|V|b>

Two-body Hamiltonians <ab|V|cd>_{JPTz}

Three-body … (ignored in IMSRG(2) truncation)



111 $$[A^{(1)}, B^{(1)}]^{(1)} = \sum_{ij}\sum_{a} :a_i^\dagger a_j: \left(A_{ia}B_{aj} - B_{ia}A_{aj}\right)$$

110 $$[A^{(1)}, B^{(1)}]^{(0)} = \sum_{ij} A_{ij}B_{ji}(n_i - n_j)$$

122 $$[A^{(1)}, B^{(2)}]^{(2)} = \frac{1}{4}\sum_{ijkl}\sum_{a} :a_i^\dagger a_j^\dagger a_l a_k: \left\{(1-P_{ij})A_{ia}B_{ajkl} - (1-P_{kl})A_{ak}B_{ijal}\right\}$$

121 $$[A^{(1)}, B^{(2)}]^{(1)} = \sum_{ij}\sum_{ab} :a_i^\dagger a_j: \left\{(n_a - n_b)A_{ab}B_{biaj}\right\}$$

222 $$[A^{(2)}, B^{(2)}]^{(2)} = \frac{1}{4}\sum_{ijkl}\sum_{ab} :a_i^\dagger a_j^\dagger a_l a_k: \left\{\frac{1}{2}(A_{ijab}B_{abkl} - B_{ijab}A_{abkl})(1 - n_a - n_b) \right.$$
$$\left. + (n_a - n_b)(1 - P_{ij} - P_{kl} + P_{ij}P_{kl})A_{aibk}B_{bjal}\right\}$$

221 $$[A^{(2)}, B^{(2)}]^{(1)} = \frac{1}{2}\sum_{ij}\sum_{abc} :a_i^\dagger a_j^\dagger: \left(A_{ciab}B_{abcj} - B_{ciab}A_{abcj}\right)\left(\bar{n}_a\bar{n}_b n_c + n_a n_b \bar{n}_c\right)$$

220 $$[A^{(2)}, B^{(2)}]^{(0)} = \frac{1}{4}\sum_{ijkl} n_i n_j \bar{n}_k \bar{n}_l \left(A_{ijkl}B_{klij} - B_{ijkl}A_{klij}\right)$$

How can we accelerate…

c.f. talk by

- Jacob Davison, TRIUMF workshop 2023
- H. Hergert, INT Program 21r-1c "Tensor Networks in Many Body and Quantum Field Theory", 2023

IMSRG-Net: data-driven approach to solve IMSRG

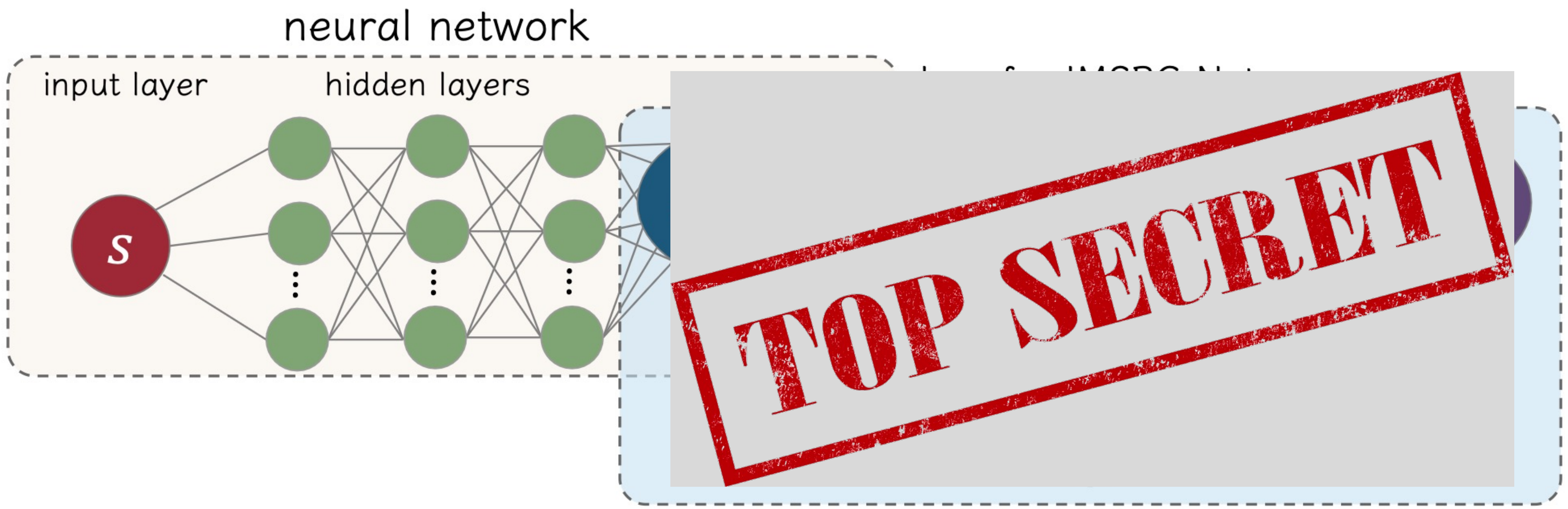
One may expect… "If we feed neural networks many data, they will learn underlying law".

It is usually not the case. We need some "inductive biases" or more constraints.

IMSRG-Net: A machine-learning based solver for In-Medium Similarity Renormalization Group

Sota Yoshida[1, *]

[1] *Institute for Promotion of Higher Academic Education, Utsunomiya University, Mine, Utsunomiya, 321-8505, Japan*

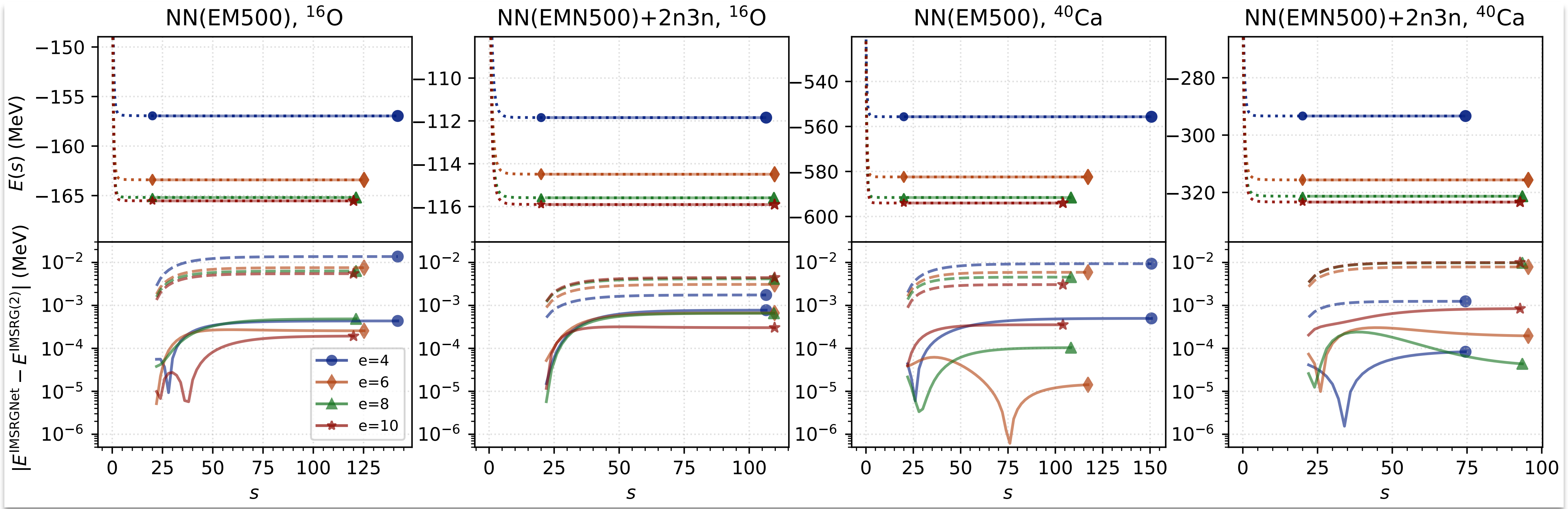


hopefully, it will be available on arXiv&GitHub within a couple of ~~days~~ weeks !!

Results: g.s. energies of ^{16}O and ^{40}Ca

"2n3n" = density dependent 3NF added

Aim is to predict E(s) or O(s) at s=∞ giving converged IMSRG(2) results



Upper panels: g.s. energy, dotted = IMSRG(2), solid = IMSRG-Net using 10 points around s=20
Lower panels: energy diff., solid = IMSRG-Net, dashed = naïve(?) ANN

Symbols: s = 20 or the point giving converged IMSRG(2)

prediction errors are less than 1 keV (~ 0.5 keV level)

Results: Energy and charge radii

"2n3n" = density dependent 3NF added

			Energy (MeV)				R_{ch} (fm)			
			$s=20$		$s=\infty$		$s=20$		$s=\infty$	
target	interection	$e_{\max}$	IMSRG(2)	IMSRG-Net	IMSRG(2)	IMSRG-Net	IMSRG(2)	IMSRG-Net	IMSRG(2)	IMSRG-Net
^{16}O	EM500	4	-156.9474	-156.9474	-156.96**11**	-156.96**07**	2.2578	2.2578	2.261**2**	2.261**0**
		6	-163.4079	-163.4079	-163.415**3**	-163.415**0**	2.2526	2.2526	2.254**7**	2.254**6**
		8	-165.1876	-165.187**5**	-165.19**32**	-165.19**27**	2.2482	2.2482	2.249**9**	2.249**7**
		10	-165.5309	-165.5309	-165.535**9**	-165.535**7**	2.2469	2.2469	2.248**5**	2.248**4**
	EMN500+2n3n	4	-111.8453	-111.8453	-111.84**70**	-111.84**62**	2.3600	2.3600	2.360**7**	2.360**5**
		6	-114.4895	-114.4895	-114.49**25**	-114.49**18**	2.3681	2.3681	2.369**2**	2.369**0**
		8	-115.5894	-115.5894	-115.59**30**	-115.59**25**	2.3735	2.3735	2.374**8**	2.374**6**
		10	-115.9040	-115.9040	-115.90**79**	-115.90**82**	2.3751	2.3751	2.3765	2.3765
^{40}Ca	EM500	4	-555.6791	-555.6791	-555.68**84**	-555.68**79**	2.5947	2.5947	2.595**9**	2.595**8**
		6	-582.4293	-582.4293	-582.435**0**	-582.435**1**	2.5960	2.5960	2.5967	2.5967
		8	-591.5783	-591.578**2**	-591.582**2**	-591.582**1**	2.5915	2.5915	2.5920	2.5920
		10	-594.0215	-594.0215	-594.02**42**	-594.02**46**	2.5890	2.5890	2.589**4**	2.589**5**
	EMN500+2n3n	4	-293.3474	-293.3474	-293.348**6**	-293.348**7**	2.8579	2.8579	2.8581	2.8581
		6	-315.6334	-315.6334	-315.6**411**	-315.64**09**	2.9082	2.9082	2.908**9**	2.908**8**
		8	-321.323**3**	-321.323**2**	-321.33**20**	-321.33**19**	2.920**1**	2.920**0**	2.920**9**	2.920**8**
		10	-323.35**19**	-321.35**20**	-323.36**05**	-323.36**13**	2.9252	2.9252	2.9260	2.9260

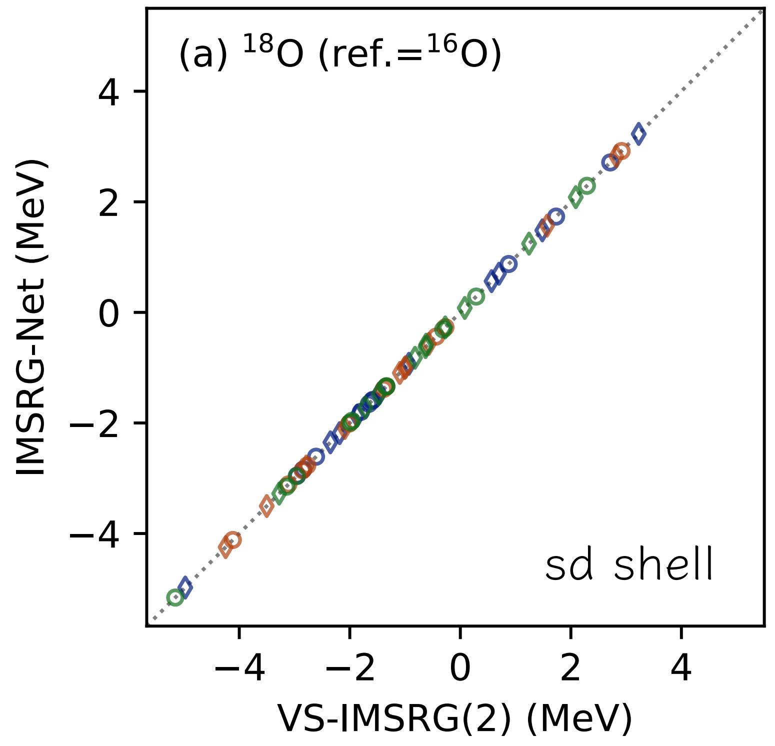
prediction errors of IMSRG-Net are much smaller than the *residuals*

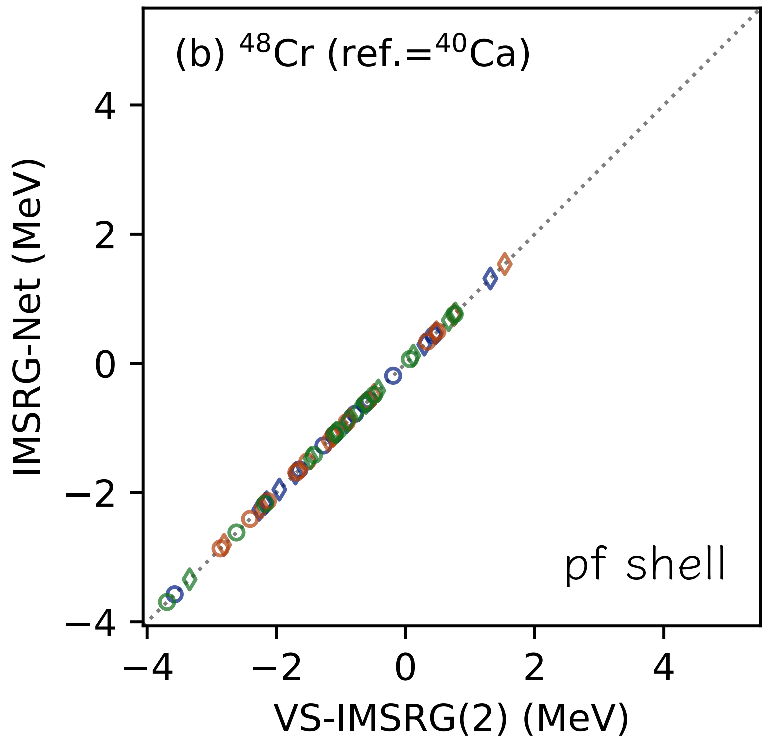
to be gained through the rest IMSRG flow (s = from 20 to ∞)

80-150

Results: valence space effective interactions

IMSRG-Net for a valence space
- same architecture and training strategy
- trained w/ earlier VS-IMSRG(2) flow



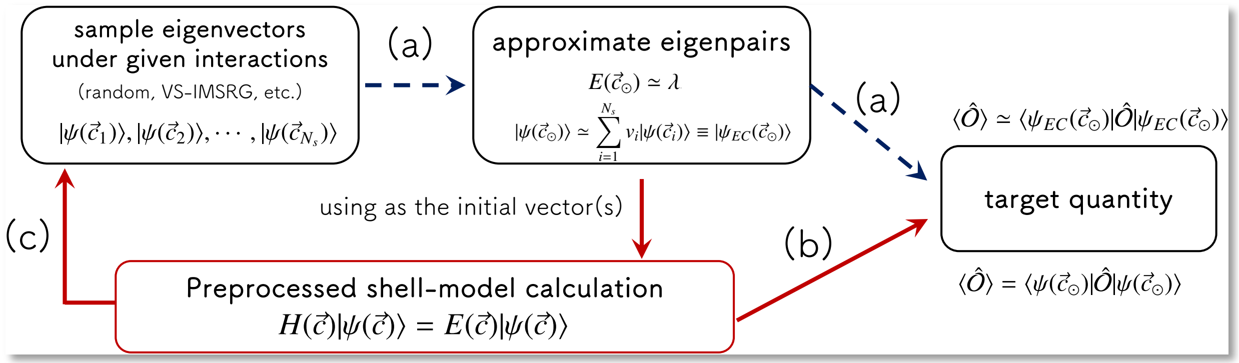


shell-model results agree in typically ≦ 1 keV level !!

※Some show ~10 keV error, attributed not to IMSRG-Net, but numerical instability of VS-IMSRG

Summary

EC + (valence) shell model


sample eigenvectors under given interactions
(random, VS-IMSRG, etc.)
$|\psi(\vec{c}_1)\rangle, |\psi(\vec{c}_2)\rangle, \cdots, |\psi(\vec{c}_{N_s})\rangle$
(a)
approximate eigenpairs
$E(\vec{c}_\odot) \simeq \lambda$
$|\psi(\vec{c}_\odot)\rangle \simeq \sum_{i=1}^{N_s} v_i|\psi(\vec{c}_i)\rangle \equiv |\psi_{EC}(\vec{c}_\odot)\rangle$
(a)
$\langle\hat{O}\rangle \simeq \langle\psi_{EC}(\vec{c}_\odot)|\hat{O}|\psi_{EC}(\vec{c}_\odot)\rangle$
target quantity
$\langle\hat{O}\rangle = \langle\psi(\vec{c}_\odot)|\hat{O}|\psi(\vec{c}_\odot)\rangle$
using as the initial vector(s)
(c)
(b)
Preprocessed shell-model calculation
$H(\vec{c})|\psi(\vec{c})\rangle = E(\vec{c})|\psi(\vec{c})\rangle$

for optimization or UQ for effective interactions

better sampling scheme is needed go beyond sd shell

SY and Noritaka Shimizu, PTEP 2022 053D02 (2022).

NuclearToolkit.jl

https://github.com/SotaYoshida/NuclearToolkit.jl
SY, Journal of Open Source Software, 7(79), 4694,(2022)

Toolkit covering chiral potentials, IMSRG, valence CI, etc.

Any feedbacks and contributions are welcomed!

IMSRG-Net Available soon

ML-based alternative of IMSRG solver. Stay tuned!!

科研費 KAKENHI

Grant: JSPS 22K14030

Transition densities

The one-body transition densities (OBTDs) is given as

$$\mathrm{OBTD}(fi; j_a j_b; \lambda) \equiv \frac{1}{\sqrt{2\lambda+1}} \langle \psi_{J_f M_f} || [c^\dagger_{j_a} \otimes \tilde{c}_{j_b}]^{(\lambda)} || \psi_{J_i M_i} \rangle, \tag{A1}$$

where we introduced $\tilde{c}_{j_b} \equiv (-1)^{j_b - m_b} c_{j_b}$, and $\langle || \cdot || \rangle$ means taking the so-called reduced matrix element, and the notation $[\cdot \otimes \cdot]^{(\lambda)}$ is for the rank-λ irreducible tensor operators. For more details on the tensor algebra, see e.g., [1, 2]. Since we are interested in the $\lambda = 0$ (scaler in terms of irreducible tensor operator) and the diagonal ($f = i$, $J_f = J_i$, $M_f = M_i$) component, which contributes to $\tilde{H}$ in Eq. (13) in the main text. $\overline{\mathrm{OBTD}}$ for the k-th single particle state is defined as

$$\overline{\mathrm{OBTD}}_k \equiv \sqrt{\frac{2j_k+1}{2J_i+1}} \mathrm{OBTD}(ii; j_k j_k; 0) = \langle \psi_{J_i M_i} || N_k || \psi_{J_i M_i} \rangle, \tag{A2}$$

where N_k is the occupation number of the k-th orbital, and the factor $\sqrt{(2j_k+1)/(2J_i+1)}$ is introduced to make $\overline{\mathrm{OBTD}}_k$ identical with the occupation number of k-th orbital.

The two-body transition densities (TBTDs) are defined as

$$\begin{aligned} &\mathrm{TBTD}(fi; abcd; J_{ab} J_{cd}; \lambda) \\ &\equiv \frac{1}{\sqrt{2\lambda+1}} \langle \psi_{J_f M_f} || [A^\dagger(ab; J_{ab} M_{ab}) \otimes \tilde{A}(cd; J_{cd} M_{cd})]^{(\lambda)} || \psi_{J_i M_i} \rangle, &\text{(A3)} \\ &\tilde{A}(cd; J_{cd} M_{cd}) \equiv (-1)^{J_{cd} + M_{cd}} A(j_c j_d; J_{cd} - M_{cd}), &\text{(A4)} \end{aligned}$$

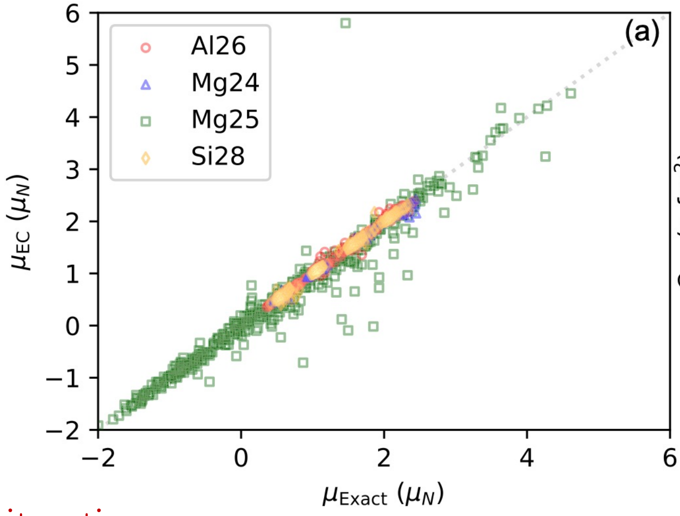
where $A^\dagger$ and A are the same as in Eqs. (5-6).For the factorization in Eq. (13). the $\overline{\mathrm{TBTD}}$ for a two-body interaction $V_J(abcd)$ is defined as follows

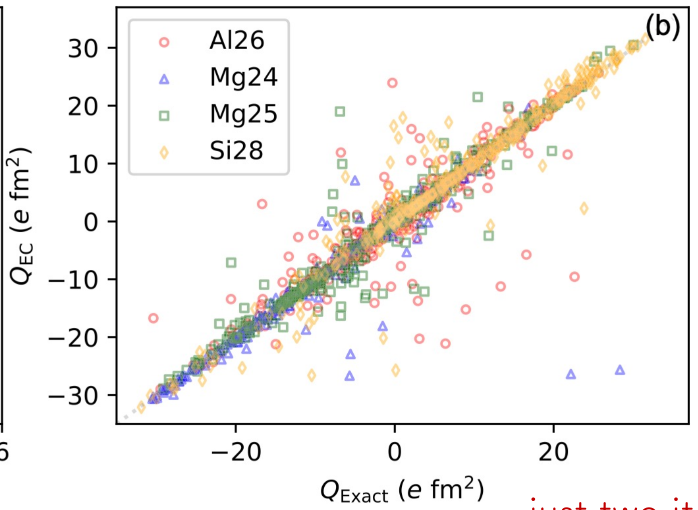
$$\overline{\mathrm{TBTD}} \equiv \sqrt{\frac{2J_{ab}+1}{2J_i+1}} \mathrm{TBTD}(fi; abcd; J_{ab} J_{ab}; 0), \tag{A5}$$

where only the term with $\lambda = 0, J_{cd} = J_{ab}, M_{cd} = M_{ab}, J_f = J_i, M_f = M_i$ is needed due to the symmetry.

μ &Q moments

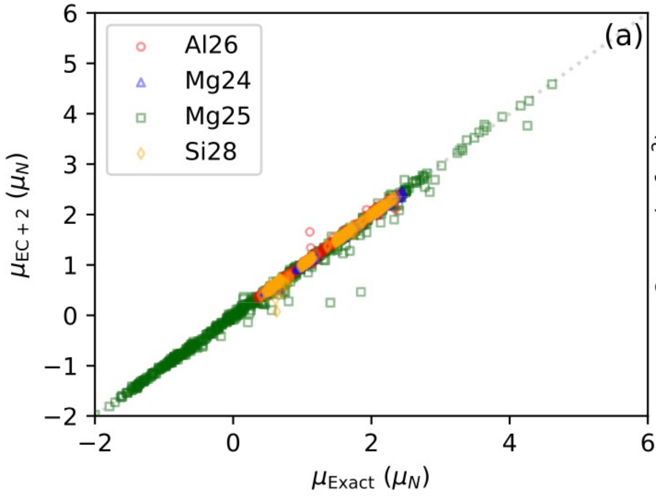
magnetic moments & quadrupole moments for the lowest states

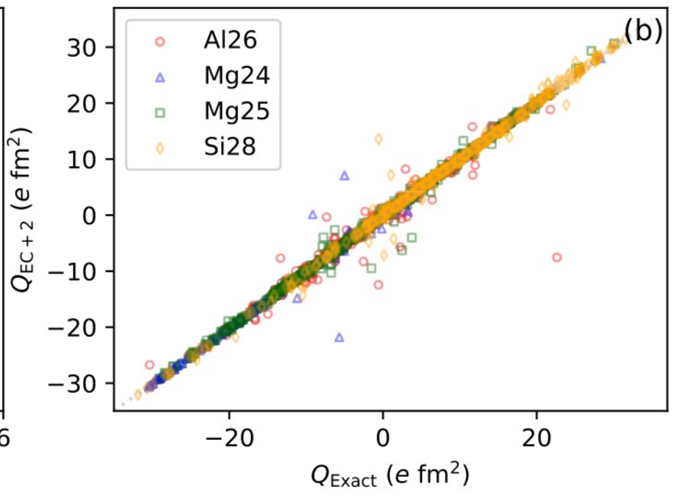




just two iterations

just two iterations





Why Julia ?

julia

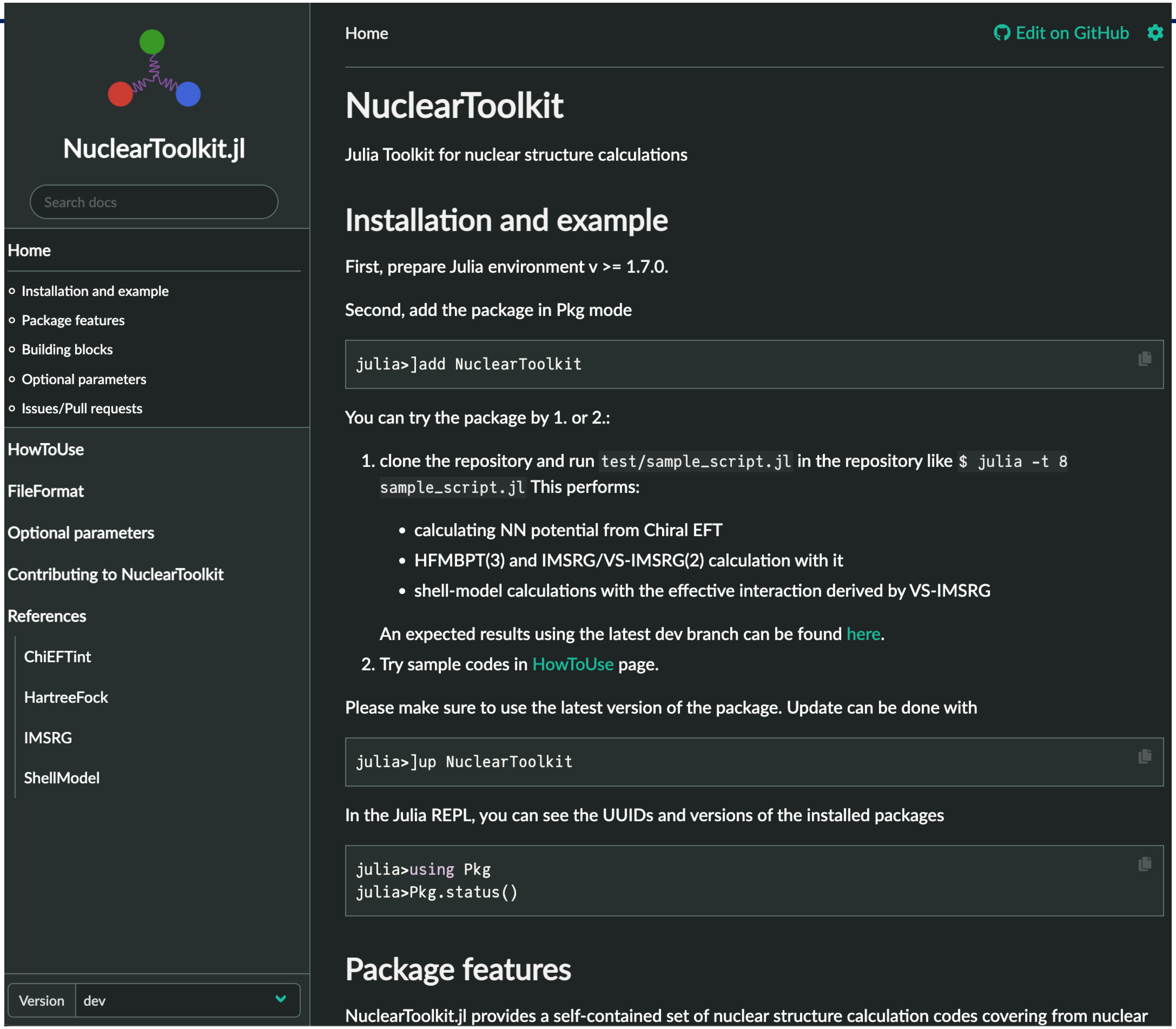
Since 2012:
Becoming popular in physics, DS, Machine Learning, etc.

- MIT LICENSE
- Multiple dispatch
- Dynamically typed
- JIT(Just-In-Time) compilation by LLVM
- Fast as C++/Fortran
- Macros like Lisp
- Package manager
- Easy to call Python, C, Fortran, etc.

→ High readability and productivity like Python
→ High performance like C++/Fortran

If you are "greedy", you should consider to use Julia 😉

NuclearToolkit.jl: How to start



Home

NuclearToolkit

Julia Toolkit for nuclear structure calculations

Installation and example

First, prepare Julia environment v >= 1.7.0.

Second, add the package in Pkg mode

```
julia>]add NuclearToolkit
```

You can try the package by 1. or 2.:

1. clone the repository and run `test/sample_script.jl` in the repository like `$ julia -t 8 sample_script.jl` This performs:
   - calculating NN potential from Chiral EFT
   - HFMBPT(3) and IMSRG/VS-IMSRG(2) calculation with it
   - shell-model calculations with the effective interaction derived by VS-IMSRG

   An expected results using the latest dev branch can be found here.
2. Try sample codes in HowToUse page.

Please make sure to use the latest version of the package. Update can be done with

```
julia>]up NuclearToolkit
```

In the Julia REPL, you can see the UUIDs and versions of the installed packages

```
julia>using Pkg
julia>Pkg.status()
```

Package features

NuclearToolkit.jl provides a self-contained set of nuclear structure calculation codes covering from nuclear