Introduction to the Jyväskylä shell model code

Jussi Toivanen

Department of Physics, University of Jyväskylä

JUSTIPEN-LACM meeting

Contents:

- Background
- Yet another shell-model code!?!
- How does the code work?
- What have we done with it so far?
- What do we expect to do with it in the future?

Background

- The FiDiPro (Finnish Distinguished Professor) programme - Jacek Dobaczewski at Jyväskylä
- One professor, two post-docs, two Ph.D students, funding for 5 years in Jyväskylä. Same also in Warzaw.
- Physics goals: Describe nuclei far from stability. Systematics of features of ground- and excited states in well and weakly bound systems
- Models: Effective field theory, new density functionals, collective models for nuclear excitations, GCM, shell model for light nuclei

Yet another shell-model code?

- J-scheme shell model code – angular momentum projected basis
- Standard Fortran 95/2003 language (top level uses advanced concepts, bottom level is fast f77 style)
- CLEAN, well documented code, modular structure, modern data structures, dynamic allocation
- Standard libraries (BLAS, LAPACK, MPI) $\Rightarrow$ Portable, "futureproof"
- *Designed* for parallel computers, should scale to $100 - 1000$s of CPUs (we hope)
- Some new ideas are used to speed things up

How does the code work?

J-scheme many-particle states:

$$|\Psi\rangle = \sum_{J_n J_p} \sum_{\alpha_n=1}^{m} \sum_{\alpha_p=1}^{n} \Psi^{J_p J_n J}_{\alpha_n \alpha_p} \left[|J_n \alpha_n\rangle |J_p \alpha_p\rangle \right]_J$$

- α_n and α_p enumerate all possible other quantum number combinations for proton and neutron states
- Dimensions small for $J = 0$ states, but grow rapidly for $J > 0$ states
- Hamiltonian matrix less sparse than in m-scheme

The hard part is the PN interaction

In projected basis the Hamiltonian is built up from rduced density matrix elements:

$$\Psi^{(IJ)K(n)}_{\alpha_n\alpha_p} = \sum_{I'J'pp'nn'\lambda} \sum_{\alpha'_n=1}^{m'} \sum_{\alpha'_p=1}^{n'} \Gamma_\lambda F(pp'nn'\lambda)(\alpha_n I||\left[c^\dagger_a \tilde{c}_{a'}\right]_\lambda||\alpha'_n I') \quad (1)$$
$$\times \quad (\alpha_p J||\left[c^\dagger_b \tilde{c}_{b'}\right]_\lambda||\alpha'_p J')\Psi^{(I'J')K(n-1)}_{\alpha'_n\alpha'_p} .$$

- Bottleneck of the calculation, scales poorly (Flops $\approx$ dimension$^{1.8}$)
- Original NATHAN-style implementation CPU-inefficient (product of lists)
- Is there a better way? YES

Double matrix product based method

The PN-hamiltonian - Lanczos vector product consists of direct matrix products, if basis ordered correctly

$$\begin{aligned}\Psi_{ij}^{(IJ)K(n)} &= \sum_{I'J'pp'nn'\lambda} \Gamma_\lambda F(pp'nn'\lambda)\left[(\mathbf{B}^{Jnn'\lambda J'} * \mathbf{A}^{Ipp'\lambda I'})\Psi^{(I'J')K(n-1)}\right]_{ij} \quad (2)\\ &= \sum_{I'J'pp'nn'\lambda} \Gamma_\lambda F(pp'nn'\lambda)\left[\sum_{i'j'} A_{ii'}^{Ipp'\lambda I'}\Psi_{i'j'}^{(I'J')K(n-1)} B_{jj'}^{Jnn'\lambda J'}\right].\end{aligned}$$

The matrices A and B are constructed from reduced density matrix elements

$$\begin{aligned}A_{ii'}^{Ipp'\lambda I'} &= (iI||\left[c_a^\dagger \tilde{c}_{a'}\right]_\lambda||i'I')\\ B_{jj'}^{Jnn'\lambda J'} &= (jJ||\left[c_b^\dagger \tilde{c}_{b'}\right]_\lambda||j'J')\,. \quad (3)\end{aligned}$$

The result:

- Scales better: Flops $\approx$ dimension$^{1.46}$, $T \approx$ dimension$^{1.28}$
- Fast: ^{56}Ni ground state in pf-shell ($dim = 1.5 \cdot 10^7$) gives 1300 sec. per Lanczos iteration (80-fold speedup)
- More efficient: Uses up to 70% of peak CPU Flops (dense matrix products - optimised BLAS)
- Works well for $J = 0$ states
- See nucl-th/0610028
- Density matrix elements stored as Variable Block Row matrices (dense blocks, some zero elements)

Used so far with ...

- Dark matter detection calculations for odd-mass nuclei ^{133}Cs, ^{127}I, ^{73}Ge, ^{83}Kr, etc.
- pf, pfg, sd+pf, sdgh model spaces
- Double beta decay studies in pf and pfg model spaces

Further improvements

Good seniority basis - work in progress

$$|\vec{n}\vec{v}\vec{\alpha}JNV\rangle = \left[\left[\left[|j_1^{n_1} v_1 \alpha_1 J_1\rangle |j_2^{n_2} v_2 \alpha_2 J_2\rangle\right]_{J_{12}} |j_3^{n_3} v_3 \alpha_3 J_3\rangle\right]_{J_{123}} |j_4^{n_4} v_4 \alpha_4 J_4\rangle\right]_J \quad (4)$$

The full many-body states are not calculated. Recursive formulas used for reduced matrix elements and Hamiltonian matrix elements.

Further improvements

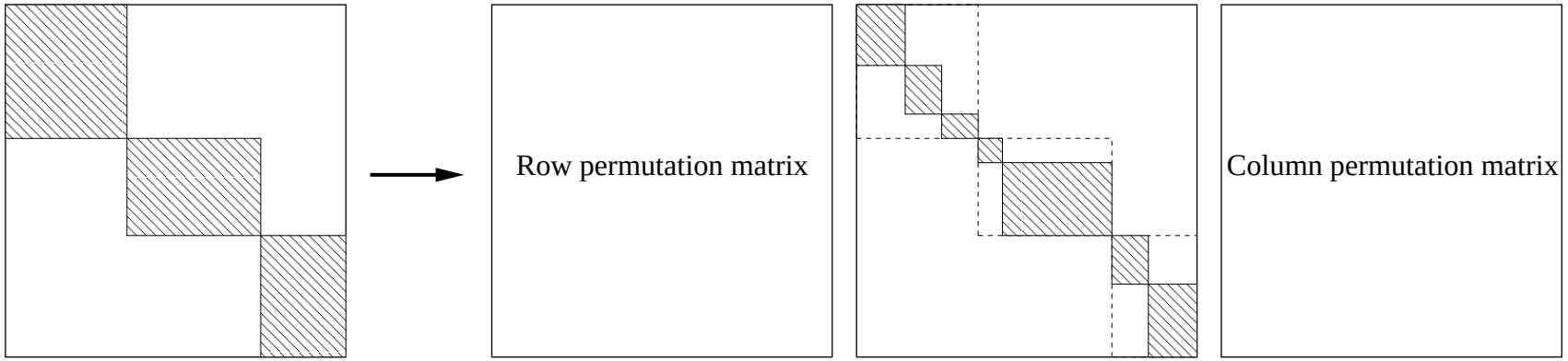
Density matrix element compression methods

$$
\begin{aligned}
(\vec{n}\vec{v}\vec{\alpha}JNV||\left[c_2^\dagger\tilde{c}_3\right]_\lambda||\vec{n}'\vec{v}'\vec{\alpha}'J'NV') &= G(j_2^{n_2}v_2\alpha_2||c_2^\dagger||j_2^{n_2-1}v_2'\alpha_2') \\
&\times (j_3^{n_3}v_3\alpha_3||\tilde{c}_3||j_3^{n_3-1}v_3'\alpha_3') \\
&\times \prod_{i\neq 2,3}\delta_{ii'}
\end{aligned}
$$

- For each operator ($c_j^\dagger$, $[c_j^\dagger\tilde{c}_{j'}]_\lambda$, $[c_j^\dagger c_{j'}^\dagger]_\lambda$) sort the basis states according to the *diagonal* quantum numbers
- $\Rightarrow$ less dense supermatrix (smaller blocks)
- good seniority also helps
- introduces permutation matrices

Further improvements

Permutation matrices



$$\mathbf{AVB}^T = \mathbf{P}_{r,a}\mathbf{A}\left(\mathbf{P}_{c,a}\mathbf{VP}_{c,b}^T\right)\mathbf{B}^T\mathbf{P}_{r,b} \tag{5}$$

Further improvements

Further use of direct matrix product structures

$$\begin{bmatrix} \mathbf{V}^{(I_1 J)K(n)} \\ \mathbf{V}^{(I_2 J)K(n)} \\ \vdots \\ \mathbf{V}^{(I_k J)K(n)} \end{bmatrix} = \sum_{\substack{I'J' \\ pp'nn'\lambda}} \Gamma_\lambda F(pp'nn'\lambda) \begin{bmatrix} \mathbf{A}^{I_1 pp'\lambda I'} \\ \mathbf{A}^{I_2 pp'\lambda I'} \\ \vdots \\ \mathbf{A}^{I_k pp'\lambda I'} \end{bmatrix} \mathbf{V}^{(I'J')K(n-1)} \left(\mathbf{B}^{Jnn'\lambda J'}\right)^T \quad (6)$$

- The result of $\mathbf{V}\,(\mathbf{B})^T$ is used for multiple $\mathbf{A}$ matrices
- $\Rightarrow$ Speeds up $J > 0$ state calculations for $N \neq Z$ nuclei

Further improvements

Combines many truncation schemes

- $N\hbar\omega$ truncations,
- seniority truncations,
- Centroid energy truncations
- Basic level and level group occupation restrictions

Exponential convergence method of Papenbrock et al. (future)

- Combining the SVD based exponential convergence method (PRC 69 024312) with the double matrix product method

Conclusion

- Code is used in Jyväskylä daily and is a testbed for new ideas in SM
- Dimensions $\approx 10^8$ in j-scheme possible now
- $\approx 10^9$ possible in the future