Building a Shell Model Code: The Pairing Model and the sd-Shell (and comparing results with NuShellX)

Gianluca Salvioni[1], Ina K. B. Kullmann[2], Matthew Shelley[3], and Gilho Ahn[4]

[1]Department of Physics, University of Jyväskylä, *gianlucasalvioni@gmail.com*
[2]Department of Physics, University of Oslo, *i.k.b.kullmann@fys.uio.no*
[3]Department of FILL IN, LaTeX University, *youremail@edu.com*
[4]Department of FILL IN, LaTeX University, *youremail@edu.com*

August 25, 2017

Abstract

We have first implemented the pairing model which have a analytical solution (to benchmark the code). Then implemented the sd shell —> more general shell-model program that allows you to study general nuclear structure problems.

developing our own shell-model code that can perform shell-model studies of the oxygen isotopes using standard effective interactions (provided by us) using as example the 1s0d shell as model space.

We have also used the NushellX code in order to perform more advanced shell-model studies and compare the results obtained with your own shell-model code to those of NushellX

and found that.....results...

I. Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

II. The pairing problem

The starting point is a pairing model: the system consists of fermions combined together in pairs of two, one with spin up and one with spin down. The assumption is that the 'breaking of pairs' is not allowed, i.e. the pairs of particles always will be coupled together, forming states with J=0. Of consequences the excited states are obtained from the excitation of two particles at the same time.

The physics of this system can be described by an Hamiltonian $\hat{H}$ consisting of an unperturbed one-body operator $\hat{H}_0$ and a two-body pertubation $\hat{V}$ defined as 'pairing potential': $\hat{H} = \hat{H}_0 + \hat{V}$. In second quantization we can write:

$$\hat{H}_0 = \sum_{p,\sigma}(\epsilon_p - 1)\hat{a}^\dagger_{p\sigma}\hat{a}_{p\sigma}, \tag{1}$$

$$\hat{V} = -\frac{1}{2}g\sum_{p,q}\hat{a}^\dagger_{p+}\hat{a}^\dagger_{p-}\hat{a}_{q-}\hat{a}_{q+}, \tag{2}$$

where the fermion creation and annihilation operators are given by $\hat{a}^\dagger_p$ and $\hat{a}_q$ respectively and $pqrs$ represent all possible single-particle quantum numbers.

The single-particle states $|p\rangle$ are chosen as eigenfunctions of the one-particle operator $\hat{h}_0$, then the Hamiltonian H acts in turn on various many-body Slater determinants constructed from the single-basis defined by the one-body operator $\hat{h}_0$.

The two-body pairing operator $\hat{V}$ is:

$$\langle q_+q_-|\hat{V}|s_+s_-\rangle = -g \tag{3}$$

where it is explicitly shown that for a given matrix element $\langle pq|\hat{V}|rs\rangle$ the states p and q (or r and s) must have opposite spin ($\sigma = \pm 1$). g is the (constant) strenght of the pairing interaction.

Using the formalism of second-quantization, the rules of anticommutation for fermions and the products of commutating and anticommutating operators can be shown that $\hat{H}_0$ and $\hat{V}$ commute with the spin projection $\hat{J}_z$ and the total spin $\hat{J}^2$. This means that H can be diagonalized in separated blocks. And due to the 'no-broken pair' assumption, only $J = 0$ are allowed.

Constructing the Hamiltonian matrix We are now interested to consider a system consisting of only four particles to calculate its exact analytic solution as benchmark for our shell-model code. The single-particle space is constructed by the four lowest levels with single-particle level energies $\epsilon_p = 1, 2, 3, 4$. Every level ϵ_p contains two particles, one with spin up $|p+\rangle$ and one with spin down $|p-\rangle$.

The possible Slater determinants that we can build are 6:

$$\begin{aligned}
|\Phi_0\rangle &= \hat{a}^\dagger_{2+}\hat{a}^\dagger_{2-}\hat{a}^\dagger_{1+}\hat{a}^\dagger_{1-}|0\rangle \\
|\Phi_1\rangle &= \hat{a}^\dagger_{3+}\hat{a}^\dagger_{3-}\hat{a}^\dagger_{1+}\hat{a}^\dagger_{1-}|0\rangle \\
|\Phi_2\rangle &= \hat{a}^\dagger_{4+}\hat{a}^\dagger_{4-}\hat{a}^\dagger_{1+}\hat{a}^\dagger_{1-}|0\rangle \\
|\Phi_3\rangle &= \hat{a}^\dagger_{3+}\hat{a}^\dagger_{3-}\hat{a}^\dagger_{2+}\hat{a}^\dagger_{2-}|0\rangle \\
|\Phi_4\rangle &= \hat{a}^\dagger_{4+}\hat{a}^\dagger_{4-}\hat{a}^\dagger_{2+}\hat{a}^\dagger_{2-}|0\rangle \\
|\Phi_5\rangle &= \hat{a}^\dagger_{4+}\hat{a}^\dagger_{4-}\hat{a}^\dagger_{3+}\hat{a}^\dagger_{3-}|0\rangle,
\end{aligned}$$

from which we can construct the basis for the Hamiltonian matrix:

$$|\Phi_0\rangle = \begin{pmatrix}1\\0\\0\\0\\0\\0\end{pmatrix}, \quad |\Phi_1\rangle = \begin{pmatrix}0\\1\\0\\0\\0\\0\end{pmatrix}, \quad \cdots \quad |\Phi_5\rangle = \begin{pmatrix}0\\0\\0\\0\\0\\1\end{pmatrix}.$$

The matrix elements $H_{ij} = \langle\Phi_i|\hat{H}|\Phi_j\rangle$ [1] are obtained using the Hamiltonian of Eq. (1), the Slater determinants and the Wick-theorem. $\hat{H}_0$ acts only on the diagonal and results in terms proportional with $(\epsilon_p - 1)$. The interaction will excite or deexcite a pair of particles from level q to level p.
Using the Wick-theorem, we notice that only the Slater determinants that differ for maximum a pair can interact among them through the pairing potential:

$$\begin{aligned}\langle\Phi_i|\hat{V}|\Phi_j\rangle &= \langle 0|\hat{a}_{i_1-}\hat{a}_{i_1+}\hat{a}_{i_2-}\hat{a}_{i_2+}\left[-\frac{1}{2}g\sum_{p,q}\hat{a}^\dagger_{p+}\hat{a}^\dagger_{p-}\hat{a}_{q-}\hat{a}_{q+}\right]\hat{a}^\dagger_{j_2+}\hat{a}^\dagger_{j_2-}\hat{a}^\dagger_{j_1+}\hat{a}^\dagger_{j_1-}|0\rangle \\ &= -\frac{1}{2}g\left(\delta_{p,i_2}\delta_{q,j_2}\delta_{i_1,j_1} + \delta_{p,i_2}\delta_{q,j_1}\delta_{i_1,j_2} + \delta_{p,i_1}\delta_{q,j_2}\delta_{i_2,j_1} + \delta_{p,i_1}\delta_{q,j_1}\delta_{i_2,j_2}\right) \qquad (4)\end{aligned}$$

and Eq.(4) is not null only if at least one of $\delta_{i_1,j_1}, \delta_{i_1,j_2}, \delta_{i_2,j_1}$ or δ_{i_2,j_2} among bra and ket states is not zero.
For example the first term in Eq.(4) is obtained from the contractions

$$\langle 0|\hat{a}_{i_1-}\hat{a}_{i_1+}\hat{a}_{i_2-}\hat{a}_{i_2+}\hat{a}^\dagger_{p+}\hat{a}^\dagger_{p-}\hat{a}_{q-}\hat{a}_{q+}\hat{a}^\dagger_{j_2+}\hat{a}^\dagger_{j_2-}\hat{a}^\dagger_{j_1+}\hat{a}^\dagger_{j_1-}|0\rangle = \delta_{p,i_2}\delta_{q,j_2}\delta_{i_1,j_1}.$$

Explicitly, the matrix element H_{00} is

$$\begin{aligned}\langle\Phi_0|\hat{H}|\Phi_0\rangle &= \langle 0|\hat{a}_{1-}\hat{a}_{1+}\hat{a}_{2-}\hat{a}_{2+}\left[\sum_{p,\sigma}(\epsilon_p - 1)\hat{a}^\dagger_{p\sigma}\hat{a}_{p\sigma}\right]\hat{a}^\dagger_{2+}\hat{a}^\dagger_{2-}\hat{a}^\dagger_{1+}\hat{a}^\dagger_{1-}|0\rangle \\ &+ \langle 0|\hat{a}_{1-}\hat{a}_{1+}\hat{a}_{2-}\hat{a}_{2+}\left[-\frac{1}{2}g\sum_{p,q}\hat{a}^\dagger_{p+}\hat{a}^\dagger_{p-}\hat{a}_{q-}\hat{a}_{q+}\right]\hat{a}^\dagger_{2+}\hat{a}^\dagger_{2-}\hat{a}^\dagger_{1+}\hat{a}^\dagger_{1-}|0\rangle \\ &= 2\cdot 0 + 2\cdot(2-1) + 2\cdot\left(-\frac{1}{2}g\right) = 2 - g. \qquad (5)\end{aligned}$$

The Hamiltonian matrix becomes

$$\hat{H} = \begin{pmatrix} 2-g & -g/2 & -g/2 & -g/2 & -g/2 & 0 \\ -g/2 & 4-g & -g/2 & -g/2 & 0 & -g/2 \\ -g/2 & -g/2 & 6-g & 0 & -g/2 & -g/2 \\ -g/2 & -g/2 & 0 & 6-g & -g/2 & -g/2 \\ -g/2 & 0 & -g/2 & -g/2 & 8-g & -g/2 \\ 0 & -g/2 & -g/2 & -g/2 & -g/2 & 10-g \end{pmatrix}. \qquad (6)$$

[1] We use the label 0 to indicate the Slater determinants with the 4 particles in the lowest states, then the first row first column matrix element is H_{00}....

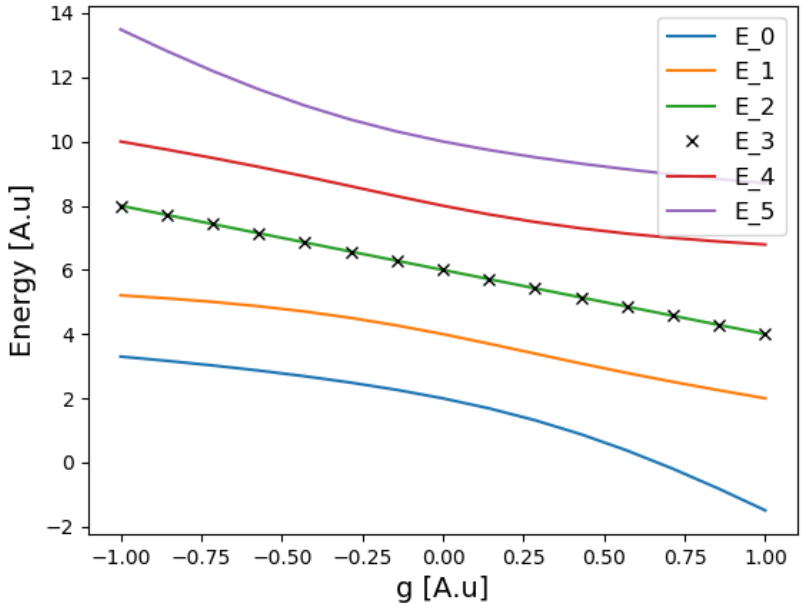

Figure 1: *The energy levels as a function of the strength g for the analytic case of the pairing model.*

In Figure 1 we see the eigenvalues of the Hamiltonian matrix (6) for the pairing model as a function of the strength $g \in [-1, 1]$. [2].

We can observe that the pairing potential increases the gap among the energy levels *E_i*, eigenvalues of Eq.(6). The case $g = 1$, corresponding to a strong attractive pairing, produces the more stable system (lowest energy levels).

i. Analytic results

are there more analytic results that we used to benchmark the code?

III. Building the shell model code

The main goal of this project was to build a working shell model code. To do this the code was built up step by step starting from the simple pairing model. A more complex system (?) was implemented when the code reproduced the analytic results from the pairing model given in section II - the code was bench-marked against known results.

develop a code which sets up the above Hamiltonian matrices for two and four particles in 2 and 4 single-particles states (the same as what you did in exercises b) and c) and obtain the eigenvalues.

What did we choose?

- Decide whether you want to read from file the single-particle data and the matrix elements in m-scheme, or set them up internally in your code. The latter is the simplest possibility

[2] The diagonalization was done with numpy

for the pairing model, whereas the first option gives you a more general code which can be extended to the more realistic cases discussed in the second part.
- Based on the single-particle basis, write a function which sets up all possible Slater determinants which have total M = 0. Test that this function reproduces the cases in b) and c). If you make this function more general, it can then be reused for say a shell-model calculation of sd-shell nuclei in the second part.
- Use the Slater determinant basis from the previous step to set up the Hamiltonian matrix.
- With the Hamiltonian matrix, you can finally diagonalize the matrix and obtain the final eigenvalues and test against the results of b) and c).

Include some of this? Or too much? :
The lecture slides contain a rather detailed recipe on how to construct a Slater determinant basis and how to set up the Hamiltonian matrix to diagonalize.

i. The structure of the code

The code is organized into blocks of code stored in separate files that performs different tasks. The files needed to run the shell model code is:

- `main.py`
- `create_table_files.py`
- `read_files.py`
- `unit_tests.py`
- `ham.py`
- `input_func.py`

The `main.py` program runs the code and calls the functions in the other files. The structure of the `main.py` program is as follows:

1. Import files
2. Ask user to provide input on the command line. Set file names
3. Create the basis vectors (vector the right word here?)
4. Create the Slater Determinants
5. Create the two-body matrix elements (tbme)
6. Obtain the total Hamiltonian matrix from the one- and two-body hamiltonian
7. Check (unit test) if the Hamiltonian for the pairing problem is correct. If not exit the program with an error message
8. Find the eigenvalues and eigenvectors of the problem with numpy

Before the basis, Slater determinants and the tbme are created the program checks, for each case if the file already exist. If the file containing for instance the basis do not exist the basis will be generated and saved to file before the code reads the file. If the file with the basis already exist the code will jump to reading the file. This way we avoid creating the (same) files every time the code is run.

To run the code one simply runs the `main.py` program in the terminal and the program will interactively ask the user for input in the terminal. The user input function is implemented in such a way that the user is not allowed to provide unphysical input values. If invalid input is given the program provides an error message and asks the user to provide the input within the allowed type or interval. When the input is given the program runs and prints the results in the terminal.

ii. Diagonalization of the Hamiltonian in the sd model space

iii. Unit tests and benchmarks

(have not done this?)

One obvious case is that of removing the two-body interaction. Then we have only the single-particle energies. For the case of degenerate single-particle orbits, that is one value of total single-particle angular momentum only j, with degeneracy $\Omega = 2j + 1$, one can show that the ground state energy E_0 is with n particles

$$E_0 = -\frac{g}{4}n(\Omega - n + 2) \tag{7}$$

Enlarge now your system to six and eight fermions and to $p = 6$ and $p = 8$ single-particle states, respectively. Run your program for a degenerate single-particle state with degeneracy Ω and test against the exact result for the ground state. Introduce thereafter a finite single-particle spacing and study the results as you vary g, as done in b) and c). Comment your results.

(describe the unit test that we have implemented and how)

iv. Challenges and proposed solutions / Future work(?)

something about the accuracy of our code somewhere here?

Example of python code

```python
import numpy as np
import sys
import matplotlib.pyplot as plt


g_list = np.linspace(-1,1,15)
N = 6
E = np.zeros(len(g_list))

plt.figure(1)
for j in range(6):
    for i in range(len(g_list)):

        g = g_list[i]

        analytical_hamiltonian = -g*np.ones([N,N])
        np.fill_diagonal(analytical_hamiltonian, 2.-2.*g)
        np.fill_diagonal(np.rot90(analytical_hamiltonian), 0)

        for n in range(N):
            analytical_hamiltonian[n,n] += 2*n

        for m in range(5,2,-1):
            analytical_hamiltonian[m,m] = analytical_hamiltonian[m-1,m-1]

        eigval, eigvec = np.linalg.eigh(analytical_hamiltonian)
```

```
27
28          E[i] = eigval[j]
29
30      if j==3:
31          plt.plot(g_list, E, 'kx')
32      else:
33          plt.plot(g_list, E)
34
35
36  plt.xlabel('g[A.u]', fontsize=14)
37  plt.ylabel('Energy[A.u]', fontsize=14)
38  plt.legend(['E_0', 'E_1', 'E_2', 'E_3', 'E_4', 'E_5'], fontsize=12)
39
40  plt.savefig('figures/eigval_vs_g.png')
41  #plt.show()
```

IV. NuShellX or NuShellX@MSU (?)

NushellX is a code developed by B. A. Brown and W. D. M. Rae (anyone else? or is it originally only Rae?)

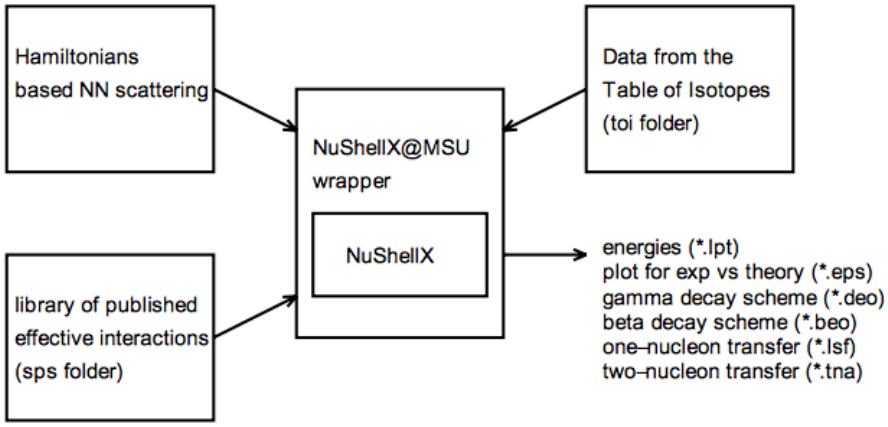

Figure 2: *Schematic layout of the NuShellX@MSU codes from ref []*

reference: https://people.nscl.msu.edu/ brown/brown-all-papers/525-2014-nds120.115-nushellx.pdf

NuShellX is a set of computer codes written by Bill Rae [1] that are used to obtain exact energies, eigenvec- tors and spectrosopic overlaps for low-lying states in shell- model Hamiltonian matrix calculations with very large basis dimensions It uses a J-coupled proton-neutron ba- sis, and J-scheme matrix dimensions of up to the order of 100 million can be considered. NuShellX@MSU is a set of wrapper codes written by Alex Brown [2] that use data files for model spaces and Hamiltonians to generate input for NuShellX. The wrap- per codes also convert the NuShellX output into figures and tables for energy levels, gamma decay and beta de- cay.

(describe what is, then put results in results section?)

V. Results

VI. Discussion

VII. Conclusions

Structure of report (crude first idea):

1. Title
2. Abstract
3. Introduction
    - Why useful
    - Where useful (nuc. chart)
    - (Look at the TALENT proposal)
    - comparing different tools
4. The Pairing problem
    - theory
    - analytic / exact results
5. Building a shell-model code
    - Describe the different parts/blocks of the code (basis, SD, states, int, hamiltonian...)
    - Unit tests / benchmarks (analytic/exact results)
    - Psedo code?
    - Accuracy of our code
    - Where our code fails, why and proposing solutions
6. NuShellX
    - describe why useful / 'powerful tool' and what may be calculated
    - compare with our own code
    - Possible 'future calculations'
7. Conclusions / Summary

References

[Github of the TALENT School] MHJ and Alex Brown *link goes here*,