Ab initio nuclear shell model calculations with a three-nucleon force for ^{15}O, ^{16}O, ^{17}O, and ^{9}Be

Hai Ah Nam

November 7, 2007

Publication Number: CSRCR2007-18

Computational Science & Engineering Faculty and Students Research Articles

Database Powered by the Computational Science Research Center Computing Group & Visualization Lab

COMPUTATIONAL SCIENCE & ENGINEERING

SAN DIEGO STATE UNIVERSITY

Computational Science Research Center
College of Sciences
5500 Campanile Drive
San Diego, CA 92182-1245
(619) 594-3430

Ab initio nuclear shell model calculations with a three-nucleon force for ^{15}O, ^{16}O, ^{17}O, and ^{9}Be

Hai Ah Nam

November 7, 2007

Faculty Advisor: Calvin W. Johnson, Department of Physics, San Diego State University
Technical Contact: W. Erich Ormand, Nuclear Theory and Modeling, N-Division,
Lawrence Livermore National Laboratory

Dissertation Proposal
San Diego State University
Claremont Graduate University

This work performed under the auspices of the U.S. Department of Energy
by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

1 EXECUTIVE SUMMARY

A reliable understanding of nuclear reactions is needed for fields as diverse as astrophysics and stewardship of the nation's weapons stockpile. When possible, scientists experimentally measure the reaction rates of interest. There are approximately 3,000 known nuclei, many of which have been produced and studied experimentally in labs. Yet, there still remains an estimated 6,000 that have yet to be created. Though new accelerator facilities allow us to study a wide range of nuclei with greater sensitivity, there are still a large number of reactions that cannot be reproduced in a laboratory.

Although we largely understand the behaviors of nuclei based on experiment and phenomenological models, a consistent and comprehensive microscopic formulation of nuclear structure, grounded in the fundamental interactions between constituent nucleons, has yet to be achieved. This is desired, to replace existing phenomenological models of nuclear structure and to be used predictively to determine nuclear properties of interest that cannot be measured experimentally, specifically transition properties needed for Reaction Theory.

Theoretical models provide the ability to study nuclei and their properties beyond the scope of what is experimentally feasible. The no-core shell model (NCSM) is one of the most successful methods of describing light nuclei ($A \leq 16$). Light nuclei studies are essential for understanding important thermonuclear and astrophysical reactions.

Due to the complex interactions within nuclei, implementing these models require the use of high-performance computing resources. Existing shell model codes are limited by run-time performance and memory constraints for large-scale calculations of interests. Codes need to be optimized, both serial and parallel portions, and unique algorithms implemented to overcome these limitations.

REDSTICK is a general utility interacting nuclear shell model program that performs NCSM calculations. Using REDSTICK and high-performance computing resources, we can perform first principles calculations of nuclear structure and reactions to address those critical gaps in our experimental knowledge. Typical shell model codes are created to handle one- and two-body interactions. Previous studies have shown that shell model calculations with only two-body forces are insufficient to capture the necessary interactions, and have been inconsistent with experimental results.

REDSTICK is one of two existing shell model codes that is capable of including three-body interactions. Although, in principle, REDSTICK can do all three-body calculations, due to the limitations of memory and run-time, significant improvements need to be made to the algorithms and parallelization of the code in order operate within these constraints.

This research will (1) focus on the algorithm and parallelization development of REDSTICK in order to (2) use first principles descriptions of nuclear structure and reactions, including three-nucleon forces for large-scale shell model calculations of light nuclei, specifically ^{15}O, ^{16}O, ^{17}O, and ^{9}Be, made possible, through our computational improvements. This research contributes to the broad effort to study nuclear structure and reactions. Success will substantially enhance the capabilities of the Nuclear Theory and Modeling Group at LLNL and DOE, through which funding has been provided for this research, and the overall national effort.

2 TECHNICAL DESCRIPTION

2.1 Scientific Motivation

A consistent microscopic formulation of the nuclear many-body problem is necessary to probe nuclear properties that cannot be measured experimentally. A complete nuclear theory is needed to describe element formation, the properties of stars and for use in present and future energy and defense applications.

Nuclear structure calculations are an important input to reactions. Light nuclei studies are essential for calculating important thermonuclear reaction rates such as ^{3}He$(\alpha,\gamma)^7$Be and ^{12}C$(\alpha,\gamma)^{16}$O, and in astrophysical reactions to improve the standard solar model, necessary for an improved understanding of neutrino oscillations. Light nuclei also provide a tractable testing ground for theoretical models, which become unwieldy as the number of particles in the nucleus increases.

Nuclear modeling has two challenges: (1) calculating the best effective interaction and single particle energies and (2) the computational difficulties of computing the energy and wavefunctions for very large scale calculations. This research will focus on the second challenge. The first shell model calculations began in the 1960s in Oak Ridge and gathered momentum in the 1980s with the development of the Oxford-Buenos Aires Shell Model (OXBASH) code for *sd*-shell nuclei. Since then, advances in computational resources and nuclear theory allow us to study nuclei with an *ab initio* approach using the NCSM framework.

Previous shell model calculations of light nuclei have been limited to two-body interactions due to the computational difficulties of higher body calculations. Results from two-body calculations, though much improved due to higher precision interactions, still remain inconsistent with experiment. In order to accurately characterize nuclei, a 3-body force is essential in the calculations. Recent advances in theory and computational power allows for the 3-body interactions in the NCSM. There have been systematic studies of the effects of three-body interactions on *p*-shell nuclei up to ^{13}C in the $4\hbar\Omega$ and $6\hbar\Omega$ model spaces with favorable results.

I propose to continue the systematic study of light nuclei in the *p*-shell and low *sd*-shell through investigations of ^{15}O, ^{16}O, ^{17}O, and ^{9}Be. These studies have not been possible due to the computational limits imposed on nuclear shell model codes. Studies will include nuclear structure calculations with 3-body forces and the effects on spin-observables.

2.2 Background

NUCLEAR SHELL MODEL. The *ab initio* no-core shell model (NCSM) [1] is one of the most successful methods used to solve the nuclear structure problem for light nuclei ($A \leq 16$). It treats the atomic nucleus as a system of A particles interacting by realistic internucleon forces. In the past, studies were limited to only two-nucleon interactions. However, recent theoretical advancements in deriving a three-body effective interaction [2], [3], [4] and the ability to solve a three-nucleon system using the NCSM approach [5] allows us

to use the three-nucleon interaction (TNI) in the NCSM Hamiltonian for p-shell nuclei (see Figure 1).

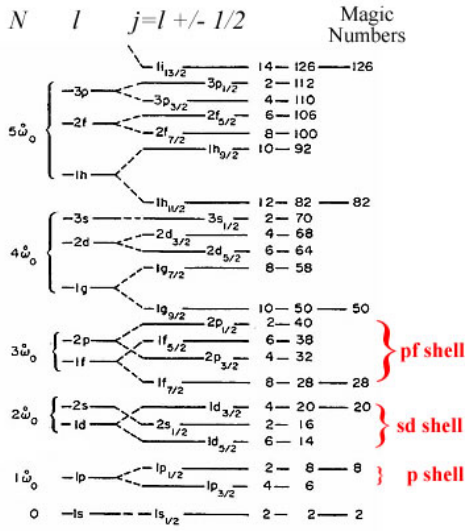


Figure 1: Schematic representation of the shell structure in nuclei up to 5 harmonic oscillator levels.

The NCSM assumes that all particles are active, leaving no inert core. In order to make the problem computationally tractable, a truncation is made to the maximum number of allowed harmonic oscillator quanta, designated by N in Figure 1, shared by all nucleons above the lowest configuration in the model space, (often called N_{max} or $N\hbar\Omega$ in literature [1]), that the particles can occupy. Typically for more accurate calculations, higher N_{max} is desired, leading to a larger model space, which results in a larger basis dimension. Thus, to perform NCSM calculations, a compromise must be made between accuracy and computational feasibility.

Calculations involve solving the Schrödinger wave equation with a basic eigenvalue problem:

$$H\psi_i = E_i\psi_i. \tag{1}$$

In second quantization formalism, the nuclear 2- and 3-body Hamiltonian look like:

$$\hat{H}_{2Body} = \sum_i \varepsilon_i a_i^+ a_i + \frac{1}{4}\sum_{ijkl} V_{ijkl} a_i^+ a_j^+ a_l a_k, \tag{2}$$

$$\hat{H}_{3Body} = \sum_i \varepsilon_i a_i^+ a_i + \frac{1}{36}\sum_{ijklmn} V_{ijklmn} a_i^+ a_j^+ a_k^+ a_n a_m a_l, \tag{3}$$

where a_i^+ is a creation operator that creates a particle in the ith location and a_i is an annihilation operator that destroys a particle in the jth location. ε_i is the single-particle

[1]The $N\hbar\Omega$ designation is also used for a tunable parameter in interaction calculations, which can be confusing. In this paper it will be used only in defining the model space.

energy and $V_{ijkl(mn)}$ are the two- or three-body interaction terms. It assumes a single particle potential and expands the full many-body solution with a convenient choice of orthogonal basis

$$\psi_i = \sum_n C_{in}\phi_n, \tag{4}$$

$$\begin{aligned} \phi &= \begin{vmatrix} \phi_i(r_1) & \phi_i(r_2) & \cdots & \phi_i(r_A) \\ \phi_j(r_1) & \phi_j(r_2) & & \phi_j(r_A) \\ \vdots & & \ddots & \vdots \\ \phi_l(r_1) & \phi_l(r_2) & \cdots & \phi_l(r_A) \end{vmatrix} \qquad (5) \\ &= a_i^+ a_j^+ \ldots a_l^+ |0\rangle. \qquad (6) \end{aligned}$$

The complete set of basis states, called Slater Determinants (SD), are constructed by distributing the A particles in all possible ways in the single-particle states. This is an optimal representation for use computationally, as each SD can be written in 1's and 0's

$$|\mathrm{SD}\rangle = (001110010...), \tag{7}$$

where each 1 corresponds to a particle occupying the single particle state and 0's correspond to an unoccupied state. The SDs are used to construct the many-body Hamiltonian matrix elements

$$H_{ij} = \langle \phi_j | \hat{H} | \phi_i \rangle, \tag{8}$$

that comprise the matrix

$$\begin{pmatrix} H_{11} & H_{12} & \cdots & H_{1N} \\ H_{21} & H_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ H_{N1} & \cdots & \cdots & H_{NN} \end{pmatrix}. \tag{9}$$

Diagonalization of the matrix using the Lanczos algorithm gives us the low-lying energy spectrum (eigenvalues). We also get a good description of the nuclear wavefunction (eigenvectors), which can be tested by computing observables and transition strengths.

The Lanczos algorithm [6]

$$\begin{array}{lllllllll} \hat{H}\mathbf{v}_1 &=& \alpha_1\mathbf{v}_1 &+& \beta_1\mathbf{v}_2 & & & & \\ \hat{H}\mathbf{v}_2 &=& \beta_1\mathbf{v}_1 &+& \alpha_2\mathbf{v}_2 &+& \beta_2\mathbf{v}_3 & & \\ \hat{H}\mathbf{v}_2 &=& & & \beta_2\mathbf{v}_2 &+& \alpha_3\mathbf{v}_3 &+& \beta_3\mathbf{v}_4 \\ \hat{H}\mathbf{v}_2 &=& & & & & \beta_3\mathbf{v}_3 &+& \alpha_4\mathbf{v}_4 &+& \beta_4\mathbf{v}_5 \end{array} \tag{10}$$

is an iterative eigenvalue solving method derived from the Arnoldi method, specifically for real symmetric matrices. It reduces the full matrix to tridiagonal form, making it easier to solve for the eigenvalues. This method is ideal for solving a large sparse matrix where only the extremum (smallest) eigenvalues are of interest. Current shell model codes require 100-200 iterations to achieve convergence for the lowest 10 eigenvalues. The primary operations include matrix-vector multiplications and vector dot products to solve for the α's and β's.

COMPUTATIONAL CHALLENGES. Despite the efficiency of the Lanczos method and the sparsity of the matrices in the shell model, performing the Lanczos operations puts a strain on the memory and run-time resources of the high-performance computing (HPC) environment. The possible investigations by shell model codes are limited by the dimensions of the Hamiltonian matrix. The dimensions grow dramatically with number of particles and valence space (model space), approximated by Eq. [11],

$$Dim \approx \begin{pmatrix} \mathrm{N}_{sps}^{p} \\ n^{p} \end{pmatrix} \begin{pmatrix} \mathrm{N}_{sps}^{n} \\ n^{n} \end{pmatrix} \longrightarrow \text{e.g.}^{60}\mathrm{Zn, in the fp-shell} = \begin{pmatrix} 20 \\ 10 \end{pmatrix} \begin{pmatrix} 20 \\ 10 \end{pmatrix} = 3.4 \times 10^{10}, \tag{11}$$

where N_{sps}^{x} is the number of single particle states in valence space for species x (proton or neutron) and n^x is the number of active particles for species x. Since we are only dealing with two- or three-body calculations, the matrices are sparse, as seen in Table 1. Matrices become more sparse with increase in basis dimension.

Nuclei (π,ν)	Model Space	proton	neutron	Dimension	Sparsity	Matrix Elements
^{20}Ne	sd	2	2	640	13.0%	532
^{24}Mg	sd	4	4	28,503	0.74%	6,011,915
^{28}Si	sd	6	6	93,710	0.34%	29,857,317
^{46}V	pf	3	3	121,440	0.36%	53,091,624
^{48}Cr	pf	4	4	1,963,461	0.04%	1,542,071,639

Table 1: Sparsity and approximate number of matrix elements using a two-body interaction for a variety of *sd*- and *pf*-shell nuclei.

Table 1 also shows the approximate number of matrix elements. Despite the increasing sparsity, there is still a considerable number of matrix elements. These values are typically calculated as `REAL(4)`, which means that it requires 4 bytes for each element. For ^{48}Cr, the over 1.5 billion matrix elements translates to approximately 6GB of memory. Although this is not unreasonable for a HPC application, for calculations of higher dimensions, the required memory goes into the terabytes (10^{12} bytes).

High Performance Computing Environment The vision evoked by the words "supercomputer" involve unlimited memory and processing power. But, this is hardly the case. The high-performance computing environment is heavily controlled by the cost of production and daily power consumption. A typical system consists of hundreds to thousands of nodes on which multiple CPUs share memory as seen in Figure 2. Thunder, the system used to

perform my calculations, ranked 34 in the Top 500 list of supercomputers in 2007 [2], consists of 1024 nodes with 4 1.4 GHz CPUs per node for a total of 4096 CPUs of processing power. Each node shares 8 GB shared memory for a total of 8192 GB of RAM. The allowed run-time

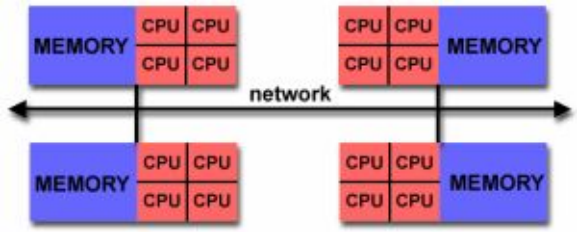


Figure 2: Shared and distributed memory architecture employed in most high-performance computing environments

is also a limiting factor in the HPC environment. The maximum time to run an application is 12 hours. There is the possibility of dedicated access time over the weekends, giving one roughly 48 hours. Additional storage can also be accessed on the parallel file systems using I/O statements. Thunder has 338 TB of memory available, but this is not allocated to one user. Usually, each user only receives a small portion of this memory.

Memory and Run-time Solutions Parallelizing a code and distributing the work amongst a large number of CPUs should reduce the overall run-time. Efficient work distribution is necessary for optimal scaling, but it is not a trivial task. Programs are also constrained by the portion of work done in serial, therefore optimized programming and algorithms are necessary, especially when working with a large-scale production code.

The pressing issue for shell model calculations are the memory constraints. To address the memory issue created by the large number of matrix elements, several approaches have been implemented in different shell model codes:

- Store all matrix elements on disk. Since many hundreds of terabytes are now typical in disk storage, this method is attractive. But, a dedicated disk space is not generally available on most HPC environments and would require a propriety system. In order to access the stored values, a very high-speed interconnect is necessary for fast I/O. Disk access is $\sim$ 1000 times slower than RAM access, resulting in higher run-times. Currently this method is used in OXBASH, Glasgow-Los Alamos, and CMICHSM.

- Store all matrix elements in RAM. This method utilizes the shared memory available on each node and is feasible in standard HPC environments. It is limited by the number of nodes one has available for the calculation. For example, 3,000 processors with 2GB of memory provides 6,000 GB RAM. As calculations become larger, the memory needs rapidly exceed the available RAM of most HPC systems. The MFD code implements this method.

[2] http://www.top500.org

- On-the-fly recomputing of the many-body matrix elements. This method only requires storage of the arrays that keep track of the non-zero matrix elements. On each iteration, the many-body matrix elements are recomputed from the two- (and three-) body matrix elements. It can put a strain on run-time due to the extra work to recompute on each iteration. The on-the-fly approach is efficient if you only compute the non-zero matrix elements. Shell model codes using this method are ANTOINE and REDSTICK (REDSTICK employs a mixture of the on-the-fly approach and storage on RAM).

3 BODY FORCES. It has been shown that the inclusion of a three-body interaction significantly improves the characterization of light nuclei [7]. Systematic studies of p-shell nuclei show that the structural impact of adding a realistic three-nucleon interaction to calculations increases the spin-orbit splitting. This results in increasing total ground state binding energy to be more consistent with experiment and improving low-lying excitation spectra (level ordering). Figure 3 (a) shows a significant improvement in the ground state energy of ^{6}Li when using the three-body interaction (AV8' + TM') over the two-body interaction (AV8') in a $6\hbar\Omega$ model space. Figure 3 (b) shows how the correct level ordering is achieved

(a)

^{6}Li basis space	Exp -	AV8'+TM'(99) $6\hbar\Omega$	AV8' $6\hbar\Omega$
$\lvert E_{gs}\rvert(1^+0)$	31.995	31.036	28.406

(c)

^{11}B→^{11}C basis space	Exp -	AV8'+TM'(99) $4\hbar\Omega$	AV8' $4\hbar\Omega$
B(GT; $\frac{3}{2}^-_1\frac{1}{2} \rightarrow \frac{3}{2}^-_1\frac{1}{2}$)	0.345	0.315	0.765
B(GT; $\frac{3}{2}^-_1\frac{1}{2} \rightarrow \frac{1}{2}^-_1\frac{1}{2}$)	0.399	0.591	0.909
B(GT; $\frac{3}{2}^-_1\frac{1}{2} \rightarrow \frac{5}{2}^-_1\frac{1}{2}$)	0.961[a]	0.517	0.353
B(GT; $\frac{3}{2}^-_1\frac{1}{2} \rightarrow \frac{3}{2}^-_1\frac{3}{2}$)	0.961[a]	0.741	0.531
B(GT; $\frac{3}{2}^-_1\frac{1}{2} \rightarrow \frac{5}{2}^-_1\frac{3}{2}$)	0.444[b]	0.625	0.197

(b)

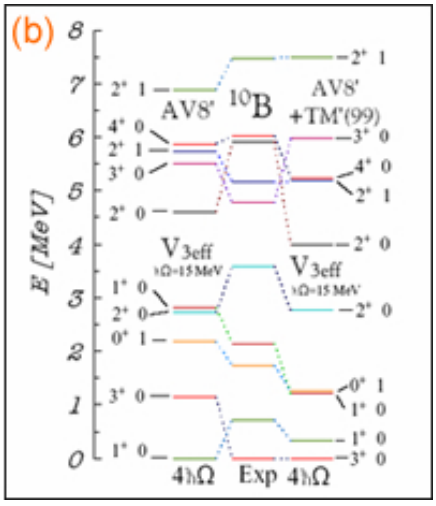


Figure 3: Effects of 3 Body Forces (a) on ground state binding energy, (b) level ordering of spectra, and (c) spin-observables (Gamow-Teller transition strengths) from Ref [7]

for the ground state and first excited state of ^{10}B by adding the three-nucleon interaction to the calculations in a $4\hbar\Omega$ space. The ordering is inverted for the two-body interaction. The addition of the TNI also improves calculations of spin-observables (e.g. magnetic moment and Gamow-Teller transition strengths) to better match experimental values. Figure 3 (c) shows closer agreement to experimental data of Gamow-Teller transition strengths for

^{11}B $\rightarrow^{11}$ C when using the three-nucleon interaction. Other results for p-shell nuclei show similar improvements using a three-nucleon interaction [7].

COMPUTATIONAL CHALLENGES OF 3 BODY FORCES. Most shell model codes perform 2-body calculations. In order to handle 3-body calculations, considerable restructuring of the code is necessary, although the algorithms can remain much the same. Assuming one has a 3-body nuclear shell model code, it still needs to be optimized and efficiently parallelized in order to handle the computational load.

The overall basis dimension for the shell model calculation does not change by going from a 2-body to a 3-body interaction. But, the calculations are more computationally intensive because the Hamiltonian matrix is far less sparse. When doing a two body calculation, the non-zero matrix elements are the result of proton-proton (p-p), neutron-neutron (n-n), and proton-neutron (p-n) interactions. For the three-body calculations, we must now consider p-p-p, n-n-n, p-p-n, and n-n-p interactions. Simple combinatorics show how choosing two items at a time is much less than choosing three items at a time:

$$\left(\begin{array}{c} 10 \\ 2 \end{array} \right) = 45 < \left(\begin{array}{c} 10 \\ 3 \end{array} \right) = 120. \tag{12}$$

The Hamiltonian matrix has far more elements, increasing the memory requirements and run-time for the calculations. For example, ^{10}B in a $4\hbar\Omega$ model space has basis dimensions of 581,740 states. The number of matrix elements and run-time to converge the lowest ten eigenvalues for:

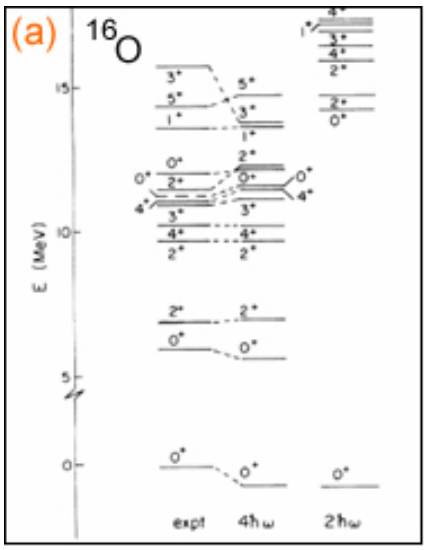
- **2 Body** are 145×10^6 non-zero elements with $\sim 1-2$ CPU-hr
- **3 Body** are 2.2×10^9 non-zero elements with ~ 200 CPU-hr

Given the substantial increase in non-zero matrix elements, it is imperative to address the algorithms and parallelization of the shell model code. Only with optimization in both areas can we achieve calculations for higher p-shell nuclei and low sd-shell nuclei.

2.3 RESEARCH

^{15}O, ^{16}O, ^{17}O, ^{9}Be. Continuing the systematic investigation of light nuclei, I propose to study the effects of three-nucleon interactions on higher p-shell nuclei (^{9}Be, ^{15}O, ^{16}O), and into the low sd-shell (^{17}O). Calculations on these nuclei with a three-nucleon interaction have not been available due to the computational inability of shell model codes to reach these dimensions in the NCSM approach. The current limit of the NCSM with a three-nucleon interaction is ^{13}C in a $6\hbar\Omega$ model space [8]. The basis dimension is roughly 32.6 million states. These calculations took roughly 4 hours on 3500 processors, or 14,000 CPU hours using the MFD shell model code.

^{16}O is a widely studied nuclei, both experimentally and theoretically. Since it is a closed-shell nuclei with eight protons and eight neutrons closing the p-shell, considered a "simpler"



(b)

Nucleus	Interaction N^3LO	CD-Bonn	V_{18}	Expt
^{15}O	6.158	6.643	4.789	7.464
^{15}N	6.339	6.810	4.957	7.699
^{16}O	6.951	7.444	5.469	7.976
^{17}O	6.722	7.201	5.214	7.751
^{17}F	6.559	7.048	5.059	7.542

Figure 4: (a) Comparison of experiment and the $4\hbar\omega$ and $2\hbar\omega$ shell model spectrum for ^{16}O with a two-body interaction, [9] (b) ground state energies for oxygen isotopes [10]

nuclear system, the shell model postulates distinct features of its structural behavior. However, two-body interactions have failed to produce shell closure evident in the divergence from experimental spectra [9].

Since ^{16}O is a closed-shell nuclei, it remains an optimal testing ground to determine whether the physics emerges from the shell model and/or realistic nuclear interactions. This is also true for ^{15}O and ^{17}O, which are 1-hole and 1-particle systems, respectively, meaning they are 1-hole or 1-particle away from a closed shell. We expect to see single particle behavior in these systems described by the shell model. Again, experimental results have not been consistent with theory. Based on the lacking theoretical results using two-body interactions only for these nuclei, seen in Figure 4, and the success of adding three-nucleon forces to other p-shell nuclei (see Figure 3), we know that the three nucleon interaction plays a critical role in determining the structure of nuclei.

^9Be provides an important test case as new experimental data will be available for comparison with theoretical results. It is believed that ^9Be is a gateway nucleus to heavier nuclei. Due to its richness of low-lying levels, understanding ^9Be will provide insight into the formation of larger nuclei. Two body results for ^9Be (see Figure 5) improve with increase in model space, but still fail to show the correct level ordering, especially for positive parity states.

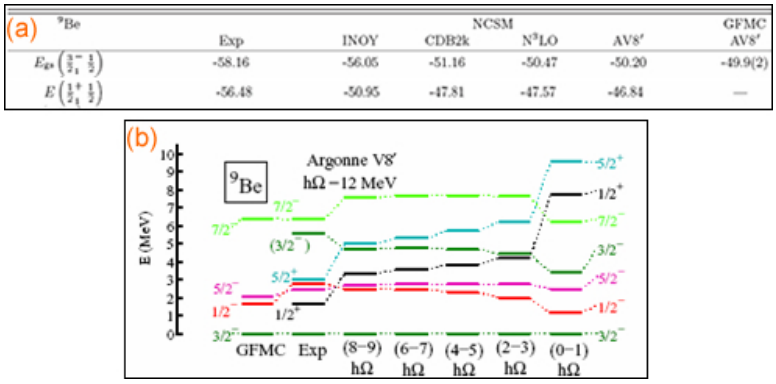


Figure 5: Two nucleon interaction results for ^{9}Be (a) ground state energies (even and odd parity) and (b) low-lying excitation spectra) [11]

.

REDSTICK. In order to do three-body calculations for these nuclei, significant optimizations need to be applied to the shell model code. The code employed in this research is REDSTICK, a general utility shell model code created by W. Erich Ormand of Lawrence Livermore National Laboratory and Calvin W. Johnson of San Diego State University. It is written in Fortran 90 and MPI, using an on-the-fly construction of the Hamiltonian matrix with an option to store some elements. The 2-body version has over 65 subroutines with over 16,000 lines of code. There exists a 3-body version. Parallelization methods are strictly confined to the Lanczos iterations, where the matrix multiplication is distributed amongst the CPUs. Standard distribution of the work for matrix multiplication can be seen in Figure 6, where each processor is given the same number of rows from the matrix to be multiplied by the vector.

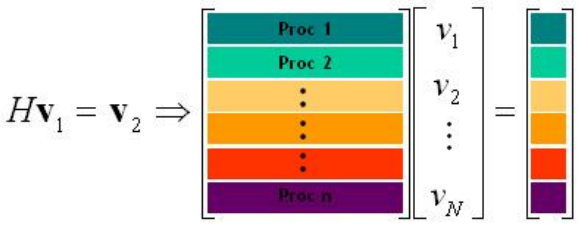


Figure 6: Standard work distribution scheme for matrix-vector multiplication

The setup of the code, involving creating the basis and establishing the necessary arrays to determine which matrix elements are non-zero, is done in serial. The utilization of on-the-fly recomputing of the matrix elements means that there are no strict memory limits for storing the matrix elements. This advantage allows REDSTICK to scale and address larger dimension problems. But, on-the-fly computing increases the run-time. Keeping the

run-time performance optimal through efficient parallelization is crucial to REDSTICK's success. General release of REDSTICK is slated for the spring of 2008.

PREVIOUS WORK. My usage and development contributions to REDSTICK are shown in the timeline below:

- **2004-2005:** Learn how to use REDSTICK for scientific investigations (input, output, physics significance).
- **2006-2007:** Implement a new Jump algorithm to various subroutines in the setup of the 2 body version. The new algorithm improves the run-time performance.
- **Summer 2007:** Create a 3-body code using the jump algorithm.

Algorithms. When first creating a program, often the first algorithms used are the most intuitive. For large production codes however, it is necessary to find the most efficient algorithms to optimize the run-time. To do on-the-fly recomputing of the matrix elements, a systematic method of finding the non-zero matrix elements is necessary. In REDSTICK, an often used procedure is determining whether two basis states connect by a one-body operator (i.e. where $\pi_i^+\pi_j$ is a one-body proton operator that removes a particle from the jth single-particle state and puts on in the ith state). If an initial and final state are found to connect by a one-body operator then the matrix element exists. For example, when finding the H^{pn} matrix elements, the contribution from one proton and one neutron interactions, the equation to find the matrix elements is:

$$H^{pn} = \sum_{ijkl} \langle \phi_f^p | \pi_i^+ \pi_j | \phi_i^p \rangle \langle \phi_f^n | \nu_l^+ \nu_k | \phi_i^n \rangle V_{ijkl}^{pn}. \tag{13}$$

Within the equation is a proton one-body jump and a neutron one-body jump. It is only when both of these are not the empty set that an H^{pn} element exists.

The previous algorithm to find these one-body matches used straight comparison. The bits from each initial state (dimension n) were compared to the bits from each final state (dimension n) to see if there was a one-particle difference. This resulted in an n^2 operation. The new algorithm I implemented uses a search routine. I start from the initial state and move the bits to unoccupied states to create all possible one-body jumps. Then using these potential final states, I search for them in the list of final states using a bisection search. Search is a $\log(n)$ operation, making the new "jump" algorithm a $n\log(n)$ operation which is a substantial improvement as seen in Figure 7, especially as the dimension size grows larger.

Parallel Performance Analysis. Using the Tuning and Analysis Utility (TAU), a portable profiling and tracing toolkit for performance analysis of parallel programs [3], we

[3] http://www.cs.uoregon.edu/research/tau

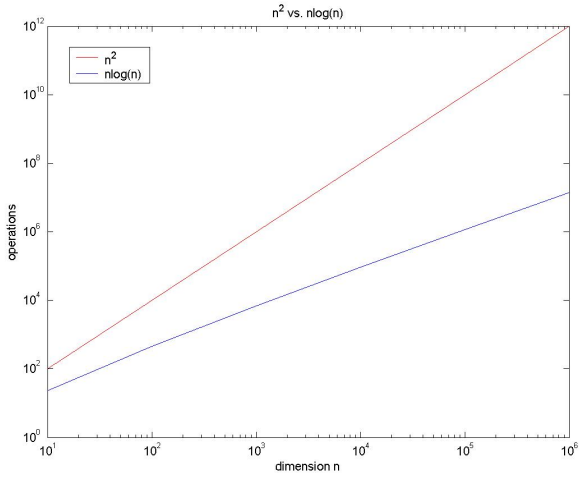


Figure 7: Dimension (n) vs. operation count for an n^2 and $n\log(n)$ operation.

can determine the run-times for the various subroutines in REDSTICK on each processor. Figure 8 shows a bar graph of the relative run-times for ^{11}C in a $5\hbar\Omega$ model space on 32 processors. The run-times for the subroutine `apply_pn` are unequal for across the different processors, thus affecting the run-times for the `MPI_Barrier` call. The unequal run-times for each processor means that the overall run-time is less than optimal, and can be improved by distributing the work more efficiently.

The natural work distribution scheme for the matrix multiplication is to give each processor an equal number of basis states. Additional analysis, shown in Figure 9, highlights how some states require several orders of magnitude more flops for their matrix multiplication workload. To avoid a workload bottleneck, and ensure proper scaling of the application, an advanced workload distribution method is necessary.

Analysis of the 2-body code, the foundation for the 3-body code, shows that the work distribution is unbalanced, affecting the scalability and run-time performance. Now that the 3-body code has been created and tested, additional parallel optimizations are necessary to perform the calculations for this research.

PROPOSED DEVELOPMENT. Although previous calculations could still run within the system's allotted time constraints with these parallelization inefficiencies, in order to perform the calculations for this research (see Table 2), we need to aggressively optimize REDSTICK. Along with additional performance analysis, I plan to implement several phases of additional parallelization.

1. **Multiple Instruction Multiple Data:** Rather than distributing the work of multiplying the Hamiltonian matrix $H = H^{ppp} + H^{nnn} + H^{ppn} + H^{nnp}$ to the vector $\mathbf{v}_1$ simply by the rows of the matrix to each node, we can also distribute the work by each component that forms the matrix. Thus, one MPI Group (with a different

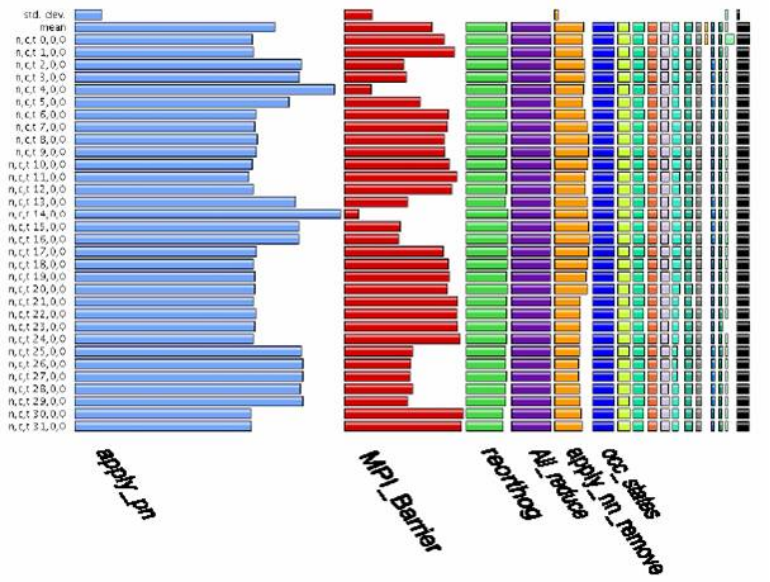


Figure 8: Bar graph showing the relative run-times broken down by subroutine on 32 processors of a 2-body REDSTICK run.

Nuclei (π,ν)	Model Space	Basis Dimension	pSD	nSD
^{15}O (7,8)	4 $\hbar\Omega$	602,455	36,369	18,999
	6 $\hbar\Omega$	34,854,023	706,318	322,795
^{16}O (8,8)	4 $\hbar\Omega$	345,365	16,812	16,812
	6 $\hbar\Omega$	26,483,625	382,612	382,612
^{17}O (8,9)	4 $\hbar\Omega$	1,517,012	4122.87	2454.29
	6 $\hbar\Omega$	105,359,444	893,586	1,426,267
^{9}Be (4,5)	4 $\hbar\Omega$	288,930	4122.87	2454.29
	6 $\hbar\Omega$	5,206,484	54,480	159,831
	8 $\hbar\Omega$	63,003,395	282,712	1,016,878

Table 2: Dimensionality for ^{15}O, ^{16}O, ^{17}O, and ^{9}Be in a variety of model spaces.

`MPI_Communicator`) with a set of nodes, will handle the multiplication for H^{ppp}, one for H^{nnn} and so forth. The benefit of allowing for each group of nodes to handle a different instruction is to alleviate the memory burden. The necessary arrays used to determine the non-zero matrix elements is also becoming quite large. The memory available to a single processor cannot handle the necessary arrays from the p-p-p, n-n-n, p-p-n, and n-n-p elements. Thus, by allowing for each group of nodes to tackle

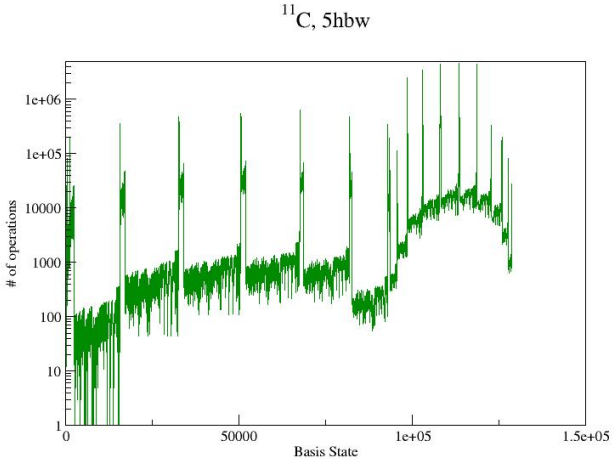


Figure 9: Proton basis index vs. the Number of floating point operations for matrix multiplication for ^{11}C in a $5\hbar\Omega$ space.

different instructions, and each node to have different data on which to perform these instructions, these calculations will be possible.

2. **Order by Jumps and Distribute Work:** In order to address the unequal run-times and idling of the processors, I will restructure the distribution of the work to the processors. Since some basis states have more non-zero matrix elements, meaning more one- and two-body jumps, by reordering the basis states by the number of jumps, we can isolate those states with more work. Then, a single processor can do the calculations for a single basis state requiring many operations, whereas several basis states with a low number of operations can be given to another processor. This will ensure and equal work distribution. The only issue we face with this is that we will always be constrained by the one state with the most number of operations. If this far exceeds the number of operations from the other states combined, we will once again see a run-time bottleneck.

3. **OpenMP and a Hybrid Programming Model:** Allowing for each processor to have a copy of the vector $\mathbf{v}_1$ on which to perform the matrix multiplication is an inefficient use of the shared memory environment. Rather, using OpenMP, a shared programming application programming interface (API), we can give each node 1 copy of the vector and allow for each processor to perform operations. The challenge here comes from ensuring that each processor is not trying to write to the same data source. This hybrid programming model will also lessen the memory requirements of the program.

ADDITIONAL IMPROVEMENTS. To push the limits on the possible calculations for the nuclear shell model, a major restructuring of existing shell model codes is necessary. As the basis dimensions grow larger, the size of a single Lanczos vector will be too large to store on a single processor. For example, basis dimensions of 10^9 requires 4GB of memory for a single vector. More advanced algorithms will be required to break up the vector across the nodes. One method could be to distribute the matrix as columns rather than rows. Since a matrix-vector multiplication is also the sum of the rows multiplied by the vector elements, this alternative approach would allow for a break up of the vector.

References

[1] P. Navratil, J. P. Vary, and B. R. Barrett. Large-basis ab initio no-core shell model and its application to ^{12}O. *Phys. Rev. C*, 62:054311, 2000.

[2] P. Navratil and B. R. Barrett. Four-nucleon shell-model calculations in a faddeev-like approach. *Phys. Rev. C*, 59:1906, 1999.

[3] P. Navratil, G.P. Kamuntavičius, and B. R. Barrett. Few-nucleon systems in a translationally invariant harmonic oscillator basis. *Phys. Rev. C*, 61:044001, 2000.

[4] P. Navratil and W. E. Ormand. Ab initio shell model calculations with three-body effective interactions for p-shell nuclei. *Phys. Rev. Lett.*, 88:152502, 2002.

[5] P. Navratil and W. E. Ormand. Feasibility study of a three-nucleon force in the no-core shell model: ^{3}H binding energy. *Phys. Rev. C*, 66:044007, 2002.

[6] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, Philadelphia, 1997.

[7] P. Navratil and W. E. Ormand. *Ab initio* shell model with a genuine three-nucleon force for the p-shell nuclei. *Phys. Rev. C*, 68:034305, 2003.

[8] P. Navratil, V. G. Gueorguiev, J. P. Vary, W. E. Ormand, and A. Nogga. Structure of $a = 10 - 13$ nuclei with two- plus three-nucleon interactions from chiral effective field theory. *Phys. Rev. Lett.*, 99:042501, 2007.

[9] W. C. Haxton and C. W. Johnson. Weak-interaction rates of ^{16}O. *Phys. Rev. Lett.*, 65:1325–1328, 1990.

[10] J. R. Gour, P. Piecuch, M. Hjorth-Jensen, M. Wloch, and D. J. Dean. Coupled-cluster calculations for valence systems around ^{16}O. *J. Phys. G*, 31:51291–51299, 2005.

[11] C. Forssén, P. Navratil, W. E. Ormand, and E. Caurier. Large basis *ab initio* shell model investigations of ^{9}Be and ^{11}Be. *Phys. Rev. C*, 71:044312, 2005.