2008

Integration of ab-initio nuclear calculation with derivative free optimization technique

Anurag Sharda
Iowa State University

Integration of ab-initio nuclear calculation with derivative free optimization technique

by

Anurag Sharda

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Masha Sosonkina, Co-major Professor
Leslie Miller, Co-major Professor
James P. Vary
Samik Basu

Iowa State University

Ames, Iowa

2008

DEDICATION

I would like to dedicate this thesis to my parents, who have been a constant source of insipiration and blessings throughout my academic career.

TABLE OF CONTENTS

LIST OF FIGURES v

ACKNOWLEDGEMENTS vii

ABSTRACT viii

CHAPTER 1 OVERVIEW 1

1.1 Introduction 1

1.2 Motivations 3

1.3 Problem Statement 6

1.4 Contributions 7

1.5 Thesis outline 8

1.5.1 Chapter 1 8

1.5.2 Chapter 2 8

1.5.3 Chapter 3 9

1.5.4 Chapter 4 9

1.5.5 Chapter 5 9

CHAPTER 2 REVIEW OF LITERATURE 10

2.1 Background work 10

2.1.1 Optimization Technique 10

2.1.2 Many Fermion Dynamic Code 20

CHAPTER 3 PROPOSED METHODS 21

3.1 Design of Integrated System 21

3.1.1 Enabling Seamless Integration 22
3.1.2 Design of χ^2 Function 26
3.1.3 Design consideration for Driver 28
3.2 Implementation 30
3.3 Alternative Design for High Performance Clusters 32
3.4 Alternative design for Grid Computing 35
3.5 Performance Monitor 38
CHAPTER 4 RESULTS 40
4.1 Platforms 40
4.2 Results 42
CHAPTER 5 CONCLUSIONS 55
5.1 Summary 55
5.2 Future Work 56
BIBLIOGRAPHY 57

LIST OF FIGURES

Figure 1.1 Matching of experimental (`Exp`) and theoretical energy levels of ^{49}Sc using CD-Bonn potential in its initial version (`CD-Bonn`) and new with searched terms (`CD-Bonn+3terms`). Each energy level is annotated with its spin value 4

Figure 1.2 Example of the scatter plot of boxes. The F-axis is function values, and the D-axis is box diameters 5

Figure 2.1 Global and local optima of a two dimensional function 13

Figure 2.2 Multiple snap shot of VTDIRECT search space at different point in time while in execution. 16

Figure 3.1 Data Flow Chart for VTDIRECT95 23

Figure 3.2 Design layout for the integration system 25

Figure 3.3 Component Level Diagram of the integration system 26

Figure 3.4 Work flow diagram for the MFDn and VTDIRECT95 supplemented with stubs 29

Figure 3.5 Design for implementing automation code for High Performance Cluster 34

Figure 3.6 Design for implementing automation code for Grid Computing . 37

Figure 4.1 Evaluations per iteration for ^{49}Ca 44

Figure 4.2 Execution times for ^{47}K, ^{48}Ca, and ^{49}Ca on different numbers of processors on Bassi 45
Figure 4.3 Matching of experimental and theoretical energy levels for ^{48}Ca . 46
Figure 4.4 Matching of experimental and theoretical energy levels for ^{49}Ca . 47
Figure 4.5 Matching of experimental and theoretical energy levels for ^{49}Ca for parallel integration code 49
Figure 4.6 Matching of experimental and theoretical energy levels for ^{47}Ca . 51
Figure 4.7 Matching of experimental and theoretical energy levels for ^{49}Ca for multiple nucleus fit 52
Figure 4.8 Matching of experimental and theoretical energy levels for ^{47}Ca for multiple nucleus fit 53

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me on the aspects of research pertaining to this thesis. First and foremost, Dr. Masha Sosonkina and Dr. James P. Vary for their guidance, motivation and support throughout the research. Weekly research meetings and discussion of the ongoing research work gave me creative insights and deep understanding of the academic material and the research field. Dr. Sosonkina helped me with planning and organizing my studies which allowed me to finish the research work on time. I greatly appreciate her for supporting me with an assistantship for my MS program. I sincerely thank my co-major professor Dr. Leslie Miller for his constant support and invaluable suggestions. I would also like to thank my committee member Dr. Samik Basu for his timely suggestions. I would also like to thank Mark Klein, Brett Bode and Troy Benjegerdes for their timely help in running simulations and configuring cluster nodes. I am thankful to Alina Negoita, Pieter Maris, Miles V. Aronnax, Andrey Shirokov and Andy Davenport for all their help and support. I would also like to extend thanks to my friends Abhishek Sinha, Abrar Hasan, Paras Malhotra, Neevan Ramalingam, Narasimha Rao and August Heim for their constant support and motivation. I would also like to acknowledge each and everyone in my research group for their participation in my research presentations.

Last but not the least, I would like to thank my parents for their constant love, support and encouragement.

ABSTRACT

Optimization techniques are finding their inroads into the field of nuclear physics calculations where the objective functions are very complex and computationally intensive. A vast space of parameters needs searching to obtain a good match between theoretical (computed) and experimental observables, such as energy levels and spectra. Manual calculation defies the scope of such complex calculation and are prone to error at the same time. This body of work attempts to formulate a design and implement it which would integrate the ab initio nuclear physics code MFDn and the VTDIRECT95 code. VTDIRECT95 is a Fortran95 suite of parallel code implementing the derivative-free optimization algorithm DIRECT. Proposed design is implemented for a serial and parallel version of the optimization technique. Experiment with the initial implementation of the design showing good matches for several single-nucleus cases are conducted. Determination and assignment of appropriate number of processors for parallel integration code is implemented to increase the efficiency and resource utilization in the case of multiple nuclei parameter search.

CHAPTER 1 OVERVIEW

1.1 Introduction

In many scientific and engineering arenas, a good combination of parameters to optimize some performance metric or cost function must be determined. Modern nuclear physics practices require use of large, sophisticated, computer intensive calculations of cost function(CF) to accurately model the interactions of a nuclei. Frequently, these CF's are not analytic, but are obtained from simulation, experiments, or from a series of numerical computation (1). Generally the CF is not smooth and multiple local optima are often present. Derivatives are usually not available in closed form, and are difficult to calculate numerically. New optimization techniques for complex cost functions have recently become an active research area, especially for global and large-region searches (2). With the increasing availability of parallel computing systems, high-performance approaches are now employed to address the high computational cost of these new optimization techniques (3),(5),(4). An optimization problem can be represented as:

Given: A function $f : A \to \mathbb{R}$ from the set A to the real numbers.

Find: An element x_0 in A such that $f(x_0) \leq f(x)$ for all x in A ("minimization") or such that $f(x_0) \geq f(x)$ for all x in A ("maximization").

Such a formulation can be called as an optimization technique or a mathematical programming problem. Optimization techniques can be broadly classified under two categories, Single Variable Optimization(SVO) and Multiple Variable Optimization(MVO). Thereafter optimization algorithms can be sub-classified as Linear Programming, Integer Programming, Heuristic Algorithms, Combinatorial Optimization, Dynamic Programming and many more. Some of the problems formulated using this technique in the domain of computer science and physics may refer to the technique as energy minimization where energy represents the value of the function f of the system begin modeled. Typically, A is some subset of the Euclidean space $\mathbb{R}^n$, often specified by a set of constraints, equalities or inequalities that the members of A have to satisfy. The domain A of f is called the search space, while the elements of A are called candidate solutions or feasible solutions. The function f is objective function, or CF. A feasible solution that minimizes (or maximizes, if that is the goal) the objective function is called an optimal solution. Generally, when the feasible region or the objective function of the problem does not present convexity, there may be several local minima and maxima, where a local minimum x^* is defined as a point for which there exists some $\delta > 0$, that for all x the expressions;

$$\|x - x^*\| \leq \delta \tag{1.1}$$

$$f(x^*) \leq f(x) \tag{1.2}$$

holds; that is to say, on some region around x^* all of the function values are greater than or equal to the value at that point. Similarly local maxima can be defined as a point in some region around x^* where all the function values are less then or equal to the value at that point. A number of algorithms proposed for solving non-convex problems including the majority of commercially available solvers are not potentially capable of making a distinction between local optimal solutions and rigorous optimal solutions, and will treat the former as actual solutions to the target problem. The branch of applied mathematics and numerical analysis that is concerned with the development of deterministic algorithms, capable of guaranteeing convergence in finite time to the actual optimal solution of a non-convex problem is called global optimization.

For twice-differentiable smooth functions, unconstrained problems can be solved by finding the stationary points where the gradient of the objective function is zero and using the Hessian matrix to classify the type of each point. If the Hessian is positive definite, the point is a local minimum, if negative definite, a local maximum, and if indefinite it is a saddle point. However, the existence of derivative is known a priori and there are methods devised for these specific situations.

The following body of work is based on integration of Derivative Free Optimization technique, DIvided RECTangle(DIRECT) Optimization algorithm and the ab inito nuclear physics Multi Fermion Dynamics(MFDn) code.

1.2 Motivations

Unlike electrons in the atom, the interaction between nucleons is not known well known and is complicated. The shell model is the fundamental tool to study the structure of nuclei. The basic idea is that the nucleons move in an average potential generated by the mutual interactions of the nucleons. The strong Nucleon-Nucleon (NN) interaction as well as 3-nucleon (NNN) interactions generate the potential that describes the nucleon energy levels in the nucleus. In particular, NN and NNN interactions tuned to fit light nuclei are used in nuclear astrophysics for solar models, supernova modeling, and Big Bang nucleosynthesis. The techniques for solving these problems also find applications in the field of quantum chemistry, condensed matter physics, atomic, nuclear, and particle physics (12).

Until recently, the No Core Shell Model (27),(28) was only applied to nuclei up to

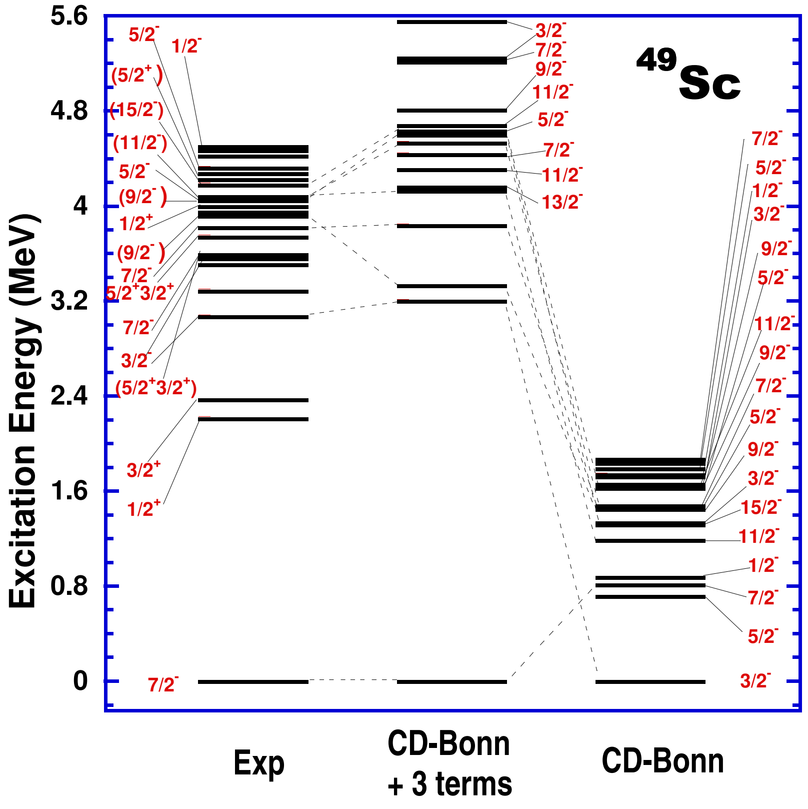

Figure 1.1 Matching of experimental (`Exp`) and theoretical energy levels of ^{49}Sc using CD-Bonn potential in its initial version (`CD-Bonn`) and new with searched terms (`CD-Bonn+3terms`). Each energy level is annotated with its spin value

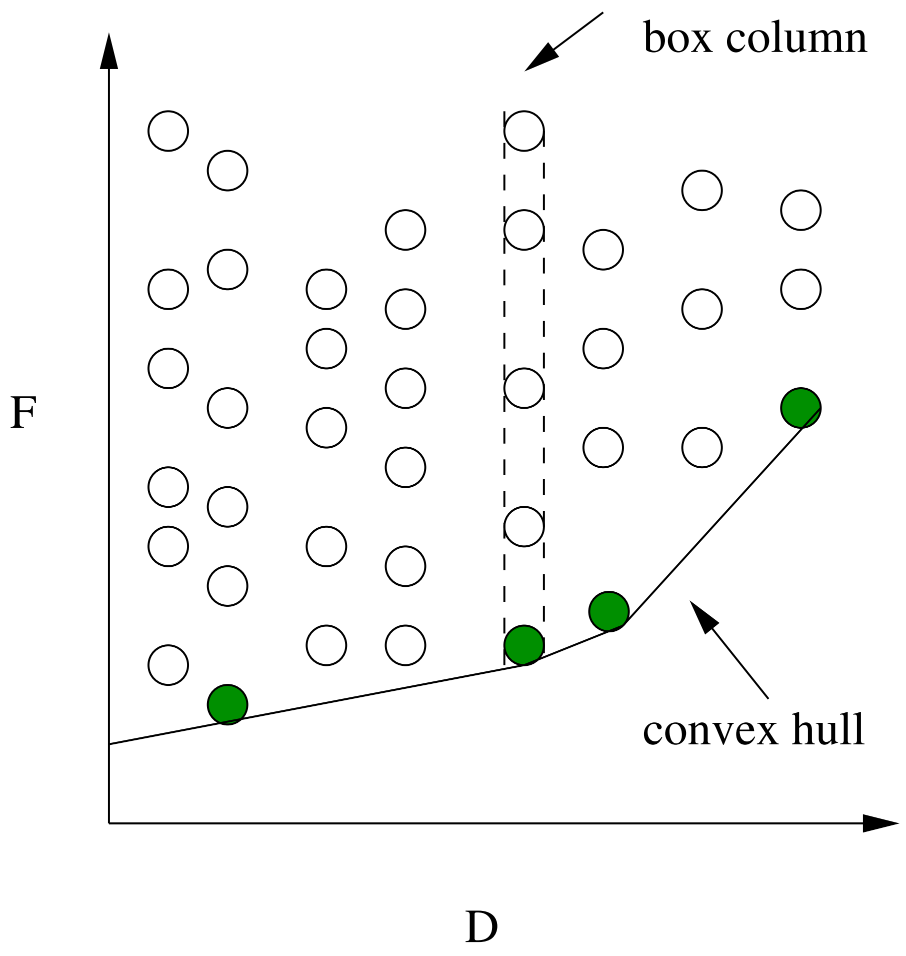

Figure 1.2 Example of the scatter plot of boxes. The F-axis is function values, and the D-axis is box diameters

atomic mass A of 16. Work is underway to extend this model to heavier nuclei (36). The effective Hamiltonian operator derived from CD-Bonn interaction (35) begins to give a poor description of nuclei with atomic mass greater then 48. Fig. 1.1 describes the matches between the theoretical and experimentally obtained energy levels for ^{49}Sc, where the initial version of the theory is given in the rightmost column. A problem with the existing Hamiltonian is that the computed spectra is too compressed compared with the experimental spectra. The addition of the three parameters isospin-dependent V_0, central V_1, and tensor-interaction V_{tens} results in a reasonable low lying spectra for the nuclei involved in the double-beta decay of ^{48}Ca. One of the physics goals is to test whether the same modified Hamiltonian described for the nuclei with atomic mass of 48 is able to describe other heavy nuclei. These three parameters and possibly other (up to 20) parameters need to be searched to obtain their best match to the experimental values (see Fig. 1.1, for example). This process is know as fitting the parameters or the fitting process. To find this match according to some criteria, it is required to evaluate energies at locus of points in a parameter search space. In particular, a criterion χ^2 may be calculated that quantifies the match using weighting (see Section 3.1.2). This process may be automated by taking advantage of optimization techniques which will generate the points at which χ^2 may be evaluated. Note that, since derivatives do not come into the picture for complex nuclear physics calculations on which χ^2 depends, the derivative-free optimization is chosen. As a trade-off, it is typically required that a large number of function evaluations are needed to find a local minimum. The time taken by such an optimization algorithm is directly proportional to the cost of the objective function evaluation. Faster processors and memory access have substantially reduced calculation time for objective functions, but parallel computing can potentially further increase efficiency,and facilitate the use of optimization techniques that were formerly considered too computationally expensive. Thus, parallel implementations of both the function evaluation and optimization algorithm may be beneficial.

1.3 Problem Statement

The process of matching theoretical and experimental values in order to find a good match typically requires large number of calculations and are thus tedious and error prone. The number of calculations increases to a much higher number when more number of parameters to be searched are introduced in the process. Each iteration of calculation of energy function requires generation of values for different parameters to be searched which can be error prone and is more complex. The probability of introduction of error and uncertainty increases dramatically when multiple nuclei are to be fitted simultaneously for same set of parameters. The automation process can take advantage of optimized technique which can generate sample points at which energy function may be evaluated. Based on these calculations more sample points are generated and thus parameter search space is explored at a rapid pace to find the minimal or the maximum function value for a given CF. Modern optimization techniques are also available in their parallel implementation . Use of these parallel implementation would reduce the searching time and thus can be more time efficient.

This body of work concentrate on implementation of a derivative free optimization technique in order to automate the process of finding the minimal energy value for the χ^2 function and design of such implementation. Implementation for both, serial and parallel version of the derivative free optimization technique is considered. To monitor and improve the performance, a performance monitor pivoted over running time of nuclei is developed which monitors running time of individual nuclei and assist in assigning corresponding number of processors to each nuclei.

1.4 Contributions

The contributions of this study are as follows:

1. Automation of the fitting process that provides following advantages:
    (a) Elimination of manual process of matching theoretical energy levels to experimental energy level and subsequent reduction in risk of any error that can be introduced in the process.
    (b) Testing and qualification of different formulation of χ^2.
    (c) Experimenting with different weighting schemes for experimental observables.

2. Design on integration of derivative-free optimization technique and large-scale application dealing with nuclear structure calculation.

3. Time Performance Monitor module to monitor the running time and performance of different nuclei on certain number of processor and to balance the resources amongst the running nuclei so as to gain the maximum resource utilization.

1.5 Thesis outline

The rest of the thesis is organized as follows

1.5.1 Chapter 1

In chapter 1, an overview on this work was provided. A brief introduction on optimization technique and ab intio nuclear calculation was given. In this chapter, the motivation behind this study was provided along with the problem statements and contributions of this study. This chapter briefly described the need of automation of fitting process used for calculations in ab initio nuclear structure and the appropriateness of derivative free optimization technique.

1.5.2 Chapter 2

In chapter 2, review of literature, related and background work for thesis is presented. It talks about various optimization techniques and their classifications in some broad categories according to properties, method of operation, number of criteria, etc. Derivative free search optimization techniques are studied in depth and in particular about VTDIRECT95 algorithm. VTDIRECT95 algorithm is studied with its implementation with different aspects of algorithm being detailed. A brief introduction to ab initio nuclear structure is presented.

1.5.3 Chapter 3

This chapter details the actual contributions of this thesis. Input/Output interfaces of VTDIRECT95 and MFDn code are discussed and detailed. Thereafter, design and implementation of serial and parallel automation code interfacing optimization technique and the target problem is presented. An alternative design for automation code that can be implemented for high performance computing platform and grid networks is also discussed. An introduction to time performance monitor module and its integration with central automation code is also presented.

1.5.4 Chapter 4

This chapter presents the results section. It details various hardware platforms that were used for developing the code and conducting experiments, different nuclei that were part of this experiment and reason for considering those nuclei. It also describes how characteristics of different nuclei (Hamiltonian sizes) affect their running time which in turn affect the number of processor allocated to avoid the bottleneck in multiple nuclei scenarios.

1.5.5 Chapter 5

This chapter talks about the conclusion and the contribution of this thesis to the field of science. It also talk about some of the future work, that can be undertaken to broaden the horizon of the current study.

CHAPTER 2 REVIEW OF LITERATURE

This section covers a brief review of literature on various optimization techniques and their classifications in some broad categories based on properties, method of operation, number of criteria and others. Derivative free search optimization techniques are studied in depth and in particular, VTDIRECT95 algorithm. VTDIRECT95 algorithm is studied with its design and implementation. Different aspects of algorithm are detailed. A brief introduction to ab initio nuclear structure is also presented.

2.1 Background work

This section covers the necessary background knowledge to understand the rest of the thesis. In the next section, related work leading to the problem statements of this thesis is explained in detail.

2.1.1 Optimization Technique

As described by Fletcher, optimization is the science of determining the best solution (6). The contribution to this is made through the mix attempts of science, statistics and heuristics, where the individuals from different fields like engineering, economics, bio-informatics and others mark a significant impact. In the following section, the state of art in the optimization technique is presented. This is not an effort to classify every possible type of optimization method.

The standard set of formulas which describe a constrained optimization problem, are seen below in 2.1.

$$
\begin{aligned}
&f \ : \ \mathbb{R}^D \ \rightarrow \ \mathbb{R} \\
&min \ f(\grave{x}) \\
&subject \ to \ \hat{g}(\grave{x}) \leq 0 \\
&h(\grave{x}) \ = \ 0
\end{aligned}
\tag{2.1}
$$

Terminology that is used and associated with the equation above are as:

- Objective/Cost Function(f) : The function to be minimized or maximized. The function need not be differentiable, or even continuous.
- Search space(ω): The part of $\mathbb{R}^D$ for which the optimum value (maximum or minimum) is to be found by optimization.
- Inequality and Equality constraints : Conditions that bounds the search space ($\hat{g}(\grave{x})$ and $h(\grave{x})$).

Optimization techniques can be categorized in many ways. The following categories are quoted from (29).

1. **Classification according to Method of Operation**

   Based on Method of Operation optimization algorithms can be divided in two main types, Deterministic and Probabilistic algorithms.

   - **Deterministic Algorithms**

     Algorithms wherein no randomness is introduced in order to generate the next candidate sampling points are known as deterministic algorithm. The characteristic of the objective function provides an obvious relationship between the possible solutions for this type of problems. Then, the search space can efficiently be explored using, for example, a divide and conquer scheme (9).

- **Probabilistic Algorithms**

  Sometimes the potential solution from the search space domain cannot be mapped directly to solve a problem. This makes it difficult and complex to solve a problem deterministically. One way to get an answer would be to apply a brute force approach, which calls for enumerating all possible candidate points in search space, which is not feasible even for relatively small problems. For such scenarios, probabilistic algorithms finds its application. Monte Carlo based approaches are one of the frequently used algorithms to answer such scenarios. Correctness of a solution are traded in for a shorter runtime of the algorithm. The decision to choose the next potential solution is made by the information gathered by the algorithm in each iteration, normally known as Heuristics.

2. **Classification according to Properties**

- **Optimization Speed**

  Based on Optimization Speed, optimization algorithms can be distinguished as two main types, Online and Offline. Some tasks are time critical, meaning that should be solved in a specified period of time. The time span can range from certain milliseconds to minutes. Problem statements falling under such constraints are known as Online optimization problems. Examples for Online optimization are robot localization, load balancing, etc. On the other hand, problem domain where time is not a critical factor are known as Offline optimization problems.

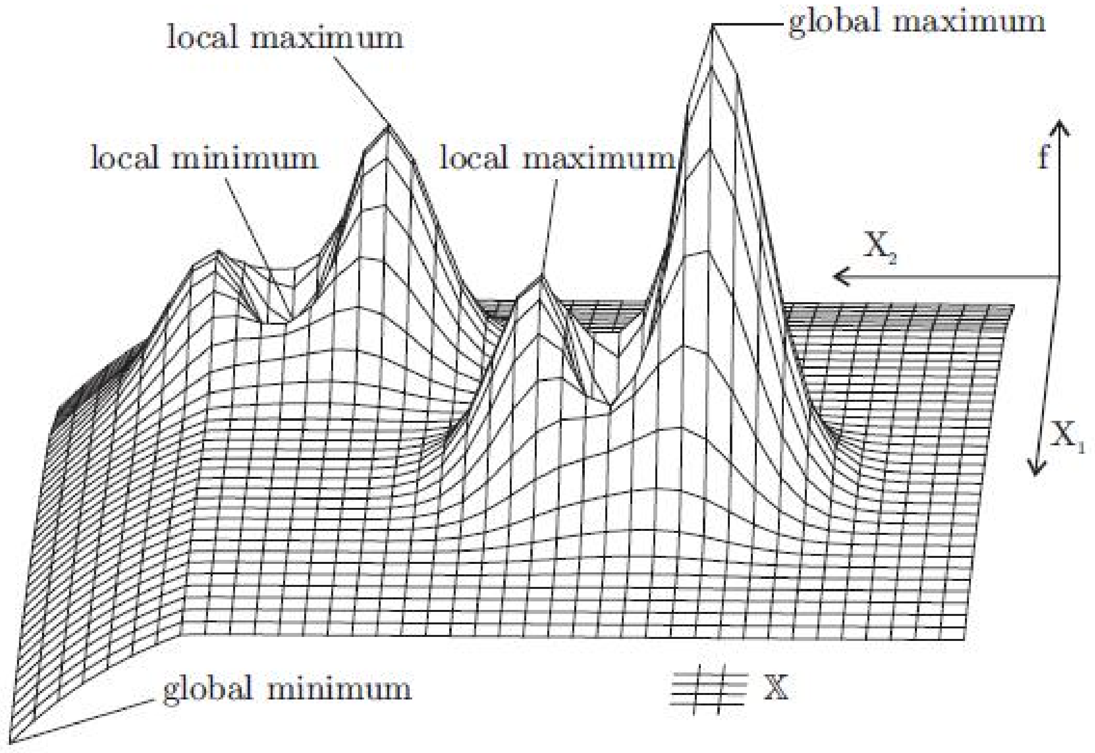

Figure 2.1 Global and local optima of a two dimensional function

- **Number of Criteria**

  Based on Number of Criteria optimization algorithms can be divided in two main categories, Single Objective Function and Multiple Objective function. While calculating single objective function best values of single objective function f is searched and in the other case a sets F of target functions is tried to optimize. The optimization for either of the case can be maximum or minimum. Algorithms designed to optimize such sets of objective functions are usually named with the prefix multi-objective.

  Figure 2.1 shows such a function f defined over a two-dimensional space X = (X1, X2). A global optimum is an optimum of the whole domain X while a local optimum is an optimum of only a subset of X. It is more likely that one wants to optimize set of multiple objective functions F, containing n =

$|F|$ objective functions f_i, each representing one criterion to be optimized.

$$F \; = \; \{f_i \; : \; X \; \rightarrow Y_i \; : \; 0 \leq i \leq n, Y_i \subseteq \mathbb{R} \; \} \tag{2.2}$$

3. **Optimization Methods**

- **Gradient Based Methods**

  For modern non linear functions, Gradient Based Methods, are primary optimization techniques. Steepest Descent Method is a classic example of gradient based minimization technique. The decision of choosing the next point is made by calculating the gradient, g , of the function at each iteration. Refer to equation 2.3:

  $$x_{k+1} \; = \; x_k \; + \; \alpha g_k \tag{2.3}$$

  The line search step alpha is added as a refinement to this very simple optimization routine. Alpha is a step size, for the sake of limiting the search step, to prevent large overshoot, or oscillations while searching for the optimum. This type of gradient-based method is a first order method, because we are using solely gradient information. Another most popular method, Newton's Method, introduces second order information in the form of Hessian, equation 2.4.

  $$x_{k+1} \; = \; x_k \; + \; \alpha {H_k}^{-1} g_k \tag{2.4}$$

  Other common derivate oriented algorithms is Sequential Quadratic Programming. Gradient based approaches are efficient and are best known methods for local optimization. The problems which this kind of methods can face are as:

  - Derivatives are not always available.

  - Finite difference approximations are too expensive or inaccurate.
  - Objective functions with various local minima or have added noise to it.
  - High dimensional problems preclude accurate estimation of gradient. (16)

- **Derivative Free Methods**

  Derivatives Free Methods were amongst the initial optimization methods. They rely on ability to compute function values and make decision based on relationship amongst the value rather then actual numeric value. The simplest one can be referred to as DIRECT Search. This algorithm checks objective function values, and accepts good points and rejects bad points, ending when a maximum iteration number has been achieved. Jones DIRECT method employs a bounding technique performing Lipschitz optimization without the Lipschitz constant (33). The advantage of these techniques is that they do not use gradients to find search directions, so in principle they can deal with noisy problems. Through the years more and more sophisticated logic has been developed to allow these types of algorithms to intelligently search through the design space. These may be as simple as distributing the search, such as in Parallel Direct Search (11), or using a simple method, as in Box's Complex Method (7).

  VTDIRECT95 (11) is a FORTRAN 95 suite of parallel code that implements derivative free optimization algorithm DIRECT. Jones et al.(10) invented DIRECT (DIviding-RECTangles) as a Lipschitzian direct search algorithm for solving global optimization problems (13),(14) subject to bound constraints of the form:

$$\min_{x \in D} \; f(x) \tag{2.5}$$

  where D = $x \in E^n | \; l \; \leq x \; \leq \; u$ is a bounded box in n-dimensional Euclidean

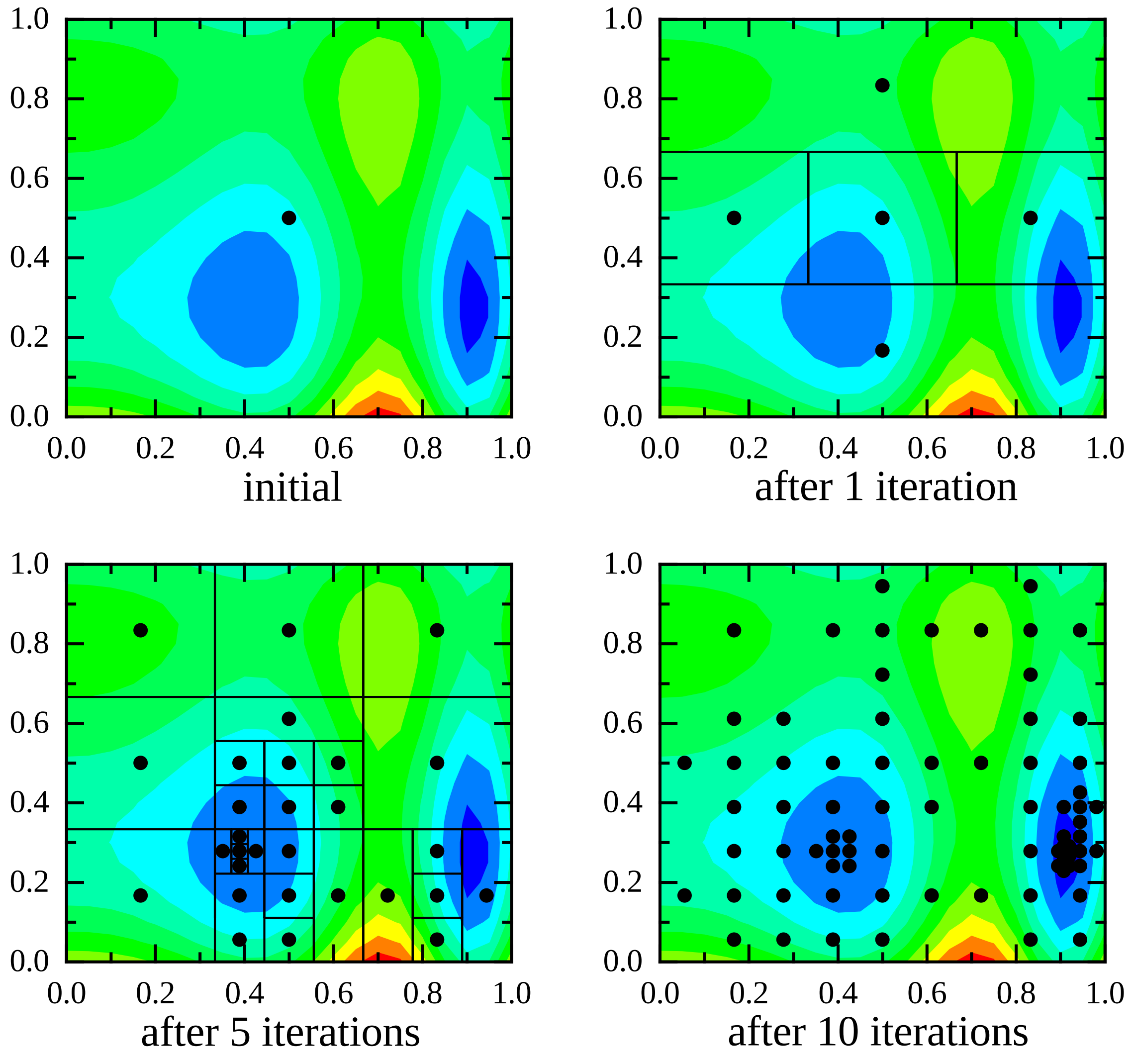

Figure 2.2 Multiple snap shot of VTDIRECT search space at different point in time while in execution.

space E^n, and f : $E^n \rightarrow E$ must satisfy a Lipschitz condition

$$|f(x_1) - f(x_2)| \leq L \ \|x_1 - x_2\|, \forall x_1, x_2 \in D. \tag{2.6}$$

VTDIRECT95 can be used for global and local optimization as it explores potentially optimal regions to converge globally for Lipschitz continuous optimization problems. As a direct pattern search method, VTDIRECT95 produces deterministic results and is straightforward to apply without derivative information or the Lipschitz constant of the objective function. It has been used successfully in many multidisciplinary design optimization problems such as high speed civil transport aircraft design (17), pipeline design (18), aircraft routing (19), surface optimization (20), wireless communication transmitter placement (21), molecular genetic mapping (22), and cell cycle modeling (24) and (25). For an objective function f inside a feasible set D, each iteration of DIRECT consists of following steps:

(a) INITIALIZATION. Normalize the feasible set D to be the unit hypercube. Sample the center point c_i of this hypercube and evaluate $f(c_i)$. Initialize $f_{\min} = f(c_i)$, evaluation counter $m = 1$, and iteration counter $t = 0$.

(b) SELECTION. Identify the set S of "potentially optimal" boxes that are subregions of D. A box is potentially optimal if, for some Lipschitz constant, the function value within the box is potentially smaller than that in any other box (a formal definition with parameter ϵ is given by (33)).

(c) SAMPLING. For any box $j \in S$, identify the set I of dimensions with the maximum side length. Let δ equal one-third of this maximum side length. Sample the function at the points $c \pm \delta e_i$ for all $i \in I$, where c is the center of the box and e_i is the ith unit vector.

(d) DIVISION. Divide the box j containing c into thirds along the dimensions in I, starting with the dimension with the lowest value of $w_i = \min\{f(c+\delta e_i), f(c-\delta e_i)\}$, and continuing to the dimension with the highest w_i. Update $f_{\min}$ and m.

(e) ITERATION. Set $S = S - \{j\}$. If $S \neq \emptyset$ go to 3.

(f) TERMINATION. Set $t = t+1$. If iteration limit or evaluation limit has been reached, stop. Otherwise, go to 2.

Initially, only one box exists in the system. As the search progresses, more boxes are generated, illustrated by the scatter plot shown in figure 1.2, where each circle represents a box. The sizes of boxes increase along the D-axis (diameter) and the function values at box centers increase along the F-axis (function). All the boxes with the same diameter belong to a box column. (33) proves that all potentially optimal boxes in S are on the lower right convex hull of the scatter plot in 1.2. To produce more tasks in parallel, new points are sampled around all boxes in S along their longest dimensions during SAMPLING. This modification also removes the step ITERATION, thus simplifying the loop. In the DIVISION step, multiple new boxes are generated for each potentially optimal box. The multiple function evaluation tasks at each iteration give rise to a natural functional parallelism, which is especially beneficial for expensive objective functions. The parallel implementation distributes the work to multiple masters in the SELECTION phase. The functions are then evaluated by the pool of workers to accomplish SAMPLING. The last step, TERMINTATION, offers a set of choices for stopping conditions. These stopping conditions are as:

(a) Maximum number of iterations : MAX ITER

(b) Maximum number of evaluations : MAX EVL (exit after a specified number of function evaluation)

(c) Minimum diameter of the box : MIN DIA (exit when the diameter of the best box has reached the value specified by the user or the round off level)

(d) Objective function convergence tolerance : OBJ CONV (exit when the relative change in the optimum objective function value has reached the given value)

One of the important characteristics of VTDIRECT95 is that it also supports user-level check pointing method to restart function evaluations through log files. Several other options are provided by the optimization algorithm to improve the performance on large-scale parallel systems.

There are numerous cases where the evaluation of the objective function is expensive and time consuming. A serial implementation of optimization technique for such categories of function would not be time and resource efficient. The parallel version of VTDIRECT95 implemented by Jian He et al (11) address such issues. The implementation takes the advantage of inherent parallelism supported by DIRECT and supports coarse-grained parallelism, which means that all the costly function evaluation are performed in parallel. A Master-Worker paradigm is supported by the parallel implementation. A single "'master"' process is responsible for calculation of potential optimal boxes, generating points and distributing this point to worker process for objective function evaluation. Each processor then calculates its own local set of potential optimal boxes and send the results back to the master process. Parallel VTDIRECT also supports dynamic load balancing which allows task migration to other processors that have finished their own task or are idle. Dynamic load balancing is achieved via a random polling algorithm using token passing (38).

2.1.2 Many Fermion Dynamic Code

Many Fermion Dynamics nuclear (MFDn) (26) parallel code is used for large-scale nuclear structure calculations in the NCSM formalism (27) (28), which has been shown to be successful for up to 16-nucleon problems on present day computational resources. MFDn code is charged to compute a few lowest (≈15) converged solutions, called wave functions, to the many-nucleon Schrödinger equation:

$$H\left|\phi\right\rangle = E\left|\phi\right\rangle . \tag{2.7}$$

Then other properties, called observables, are formed from the calculated wave functions. The matrix H in equation (2.7) is the Hamiltonian operator, which is typically solved using Lanczos diagonalization since H is symmetric and sparse. However, the Lanczos iterative process may be very expensive due to huge dimensionality of H with many off-diagonal elements. The number of Lanczos iterations also increases significantly for the energy levels beyond the ground state. For example, for the ^{16}O nucleus in the $6\hbar\omega$ basis space, the ground-state energy level requires only 35 Lanczos iterations, while 15 excited states need at least 300 Lanczos iterations for convergence. Note that, in this case, the constructed Hamiltonian H has the dimension of 26,483,625. MFDn constructs the *m-scheme* basis space, evaluates the Hamiltonian matrix elements in this basis using efficient algorithms, diagonalizes the Hamiltonian to obtain the lowest eigenvectors and eigenvalues, then post-processes the wave functions to obtain a suite of observables and to compare them with experimental values.

CHAPTER 3 PROPOSED METHODS

This chapter is divided into two parts. The first part of this chapter discusses design consideration of the automation code needed for the integration of the VTDIRECT95 code and the MFDn code. It also discusses additional design needed for high performance clusters and grid computing. The second part of the chapter concentrates on the implementation of the proposed design and the performance module.

3.1 Design of Integrated System

To get a better understanding of design goals input and output interfaces of VTDIRECT95 and MFDn are discussed and detailed. From previous chapter it can be recalled that VTDIRECT95(11) is a FORTRAN 95 suite of parallel code that implements derivative free optimization algorithm DIRECT. VTDIRECT95 normalizes the search space and evaluate the center point of a D-dimensional rectangle, where D is the number of parameters which forms a D dimensional search space. Next, it evaluates the point around the center of the rectangle, and thereafter divide the rectangle according to the function values. Lipschitz conditions and rectangle diameters are used to determine which rectangle should be divided next. Point in potentially optimal boxes are then evaluated and iteration is carried on for division and evaluation until the stopping criteria are met Fig. 3.1. VTDIRECT95 works with the notion of *iterations* and *evaluations*. In one iteration, optimization algorithm divides one or many potentially optimal boxes and samples multiple sample points. Evaluation of each such sample point is considered as one evaluation. It implies directly that one iteration can have one or more evaluations. This notion is important to understand as in serial implementation each evaluation is done one after the other, while in parallel implementation a number of evaluations can be done simultaneously, depending upon the availability of masters and workers.

MFDn code is a parallel code that is used for ab-initio nuclear calculation. MFDn code requires a set of input files for performing ab-initio calculation. This input files are the Hamiltonian files, the Interaction files and files that provide parameters set and variable values, such as number of Lanczos iteration etc., for calculation to MFDn code. The output of the MFDn code is a set a text file that have information about theoretical observables, such as excitation energies at various level, their relative nuclear spins and wave function values. This calculated excitation energy are matched against experimental energy based on the nuclear spin of each level. The number of processor required for MFDn execution typically depends on the size of the Hamiltonian matrix of the nucleus taken in consideration and hardware platform used for experiment. The typical size of Hamiltonian matrix considered for the experiments done for this work are approximately 14,000.

3.1.1 Enabling Seamless Integration

It can be observed that MFDn and VTDIRECT95 code feature typical interfaces via file I/O and function calls. In nuclear structure calculations, the computed (theoretical) and experimental results are matched for single as well as multiple nuclei using the χ^2 function. Also, the optimization techniques being considered provides a serial and a parallel interface of its implementation. By saying, serial and parallel implementation of optimization technique it is meant that in serial implementation there would be only one worker, hence all the generated sample points would be evaluated one after the other. While in parallel implementation, a number of workers would be available to evaluate

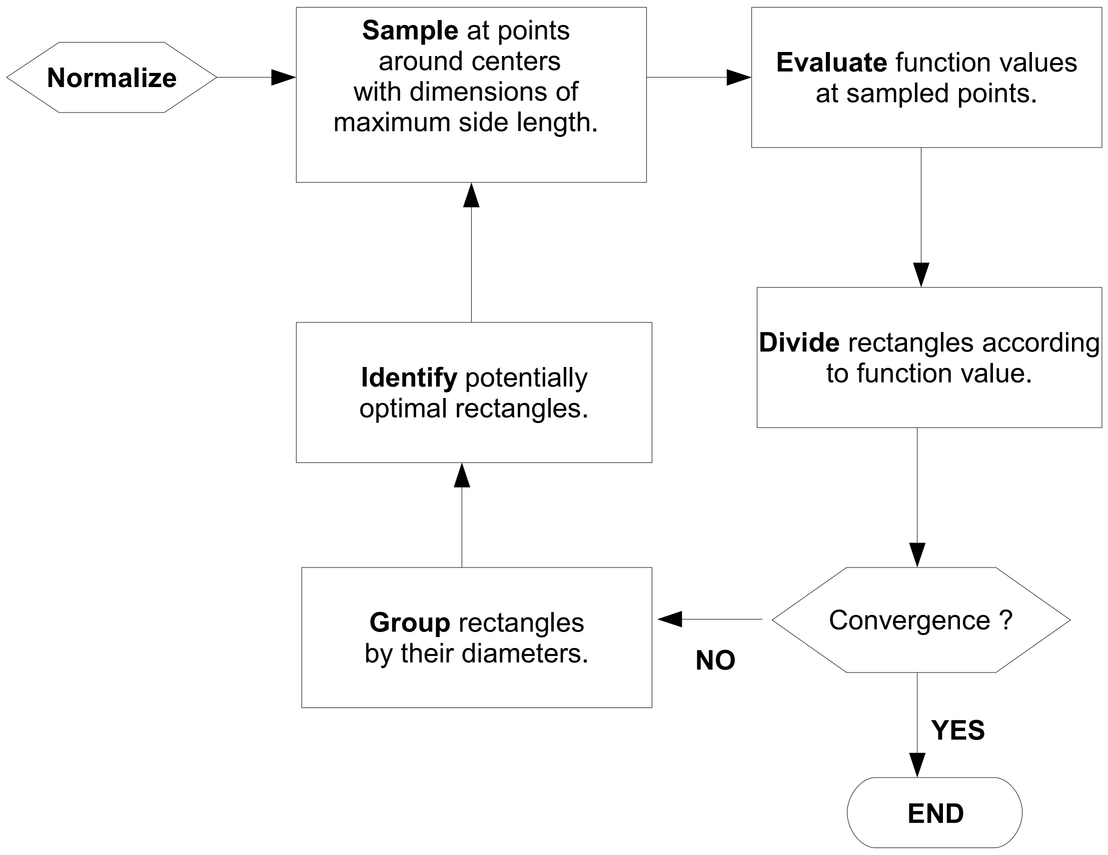
Normalize
Sample at points around centers with dimensions of maximum side length.
Evaluate function values at sampled points.
Divide rectangles according to function value.
Convergence ?
NO
YES
END
Group rectangles by their diameters.
Identify potentially optimal rectangles.

Figure 3.1 Data Flow Chart for VTDIRECT95

sample points in parallel. Thus, a general case of multiple nuclei - multiple workers is considered in the design. Figure Fig. 3.2 shows a diagram for the multiple nuclei - multiple worker case, with one MFDn execution per nucleus, and multiple sub domain masters for VTDIRECT95. A short description of design for both codes is outlined in the following subsections.

By stating seamless integration it is meant that no modification is applied to the interfaces that are defined by the packages but to use packages as 'black box' components. Package interfaces are used in such a way that the existing optimization technique can use any other problem domain and existing problem domain can use any other optimization technique. Thus it gives a loose coupling between unrelated components in the design. To create a logical flow of data, satisfy input requirement of components and produce formatted output, stubs are defined and implemented. These stubs provide a bare bone structure such that with the change in either optimization technique or problem domain, minimum work is required to connect the new set of components.

Fig. 3.2 shows a diagram for the case of multiple nuclei, N1, N2, and N3, with one MFDn execution per nucleus. Assuming that system runs in parallel mode, multiple workers are active at any given instance. Each worker would execute all the three nuclei, N1, N2, and N3, for each sample point it evaluates. The overall hierarchy consists of three tiers. The first tier consists of VTDIRECT95 process that are responsible for initialization, sampling and division for the optimization technique. Evaluation points from this tier are then passed to the workers in the next tier, which in turn spawn MFDn process, in the third tier, for evaluation of objective function. The MFDn pool of workers is static in the case of sequential automation code while for parallel automation code, this pool dynamic. The workers for MFDn pool are added and removed from the group at run time. This is due to the fact that MFDn process are spawned dynamically and they leave the group as soon as they finish evaluating the sample point.

The proposed integration system can be divided into basic three components, namely,

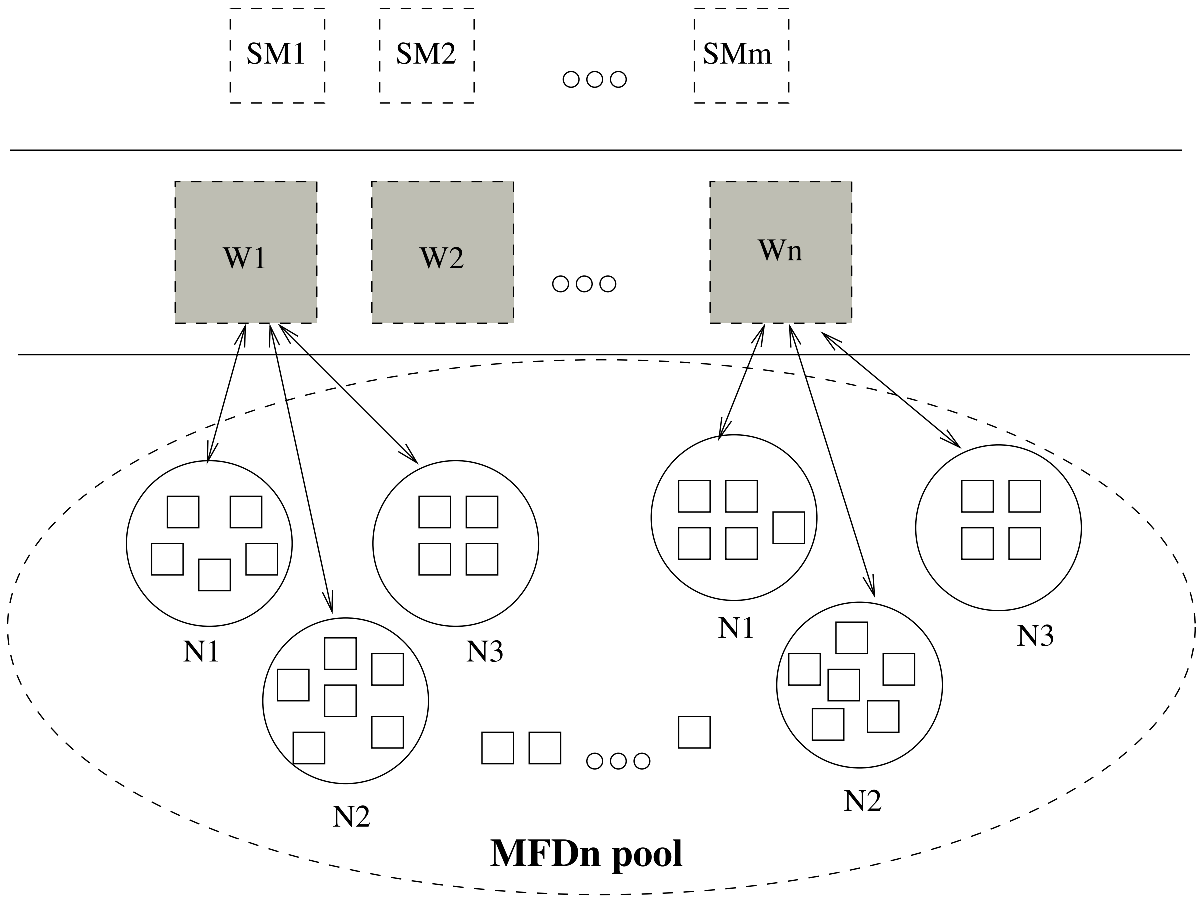

Figure 3.2 Design layout for the integration system

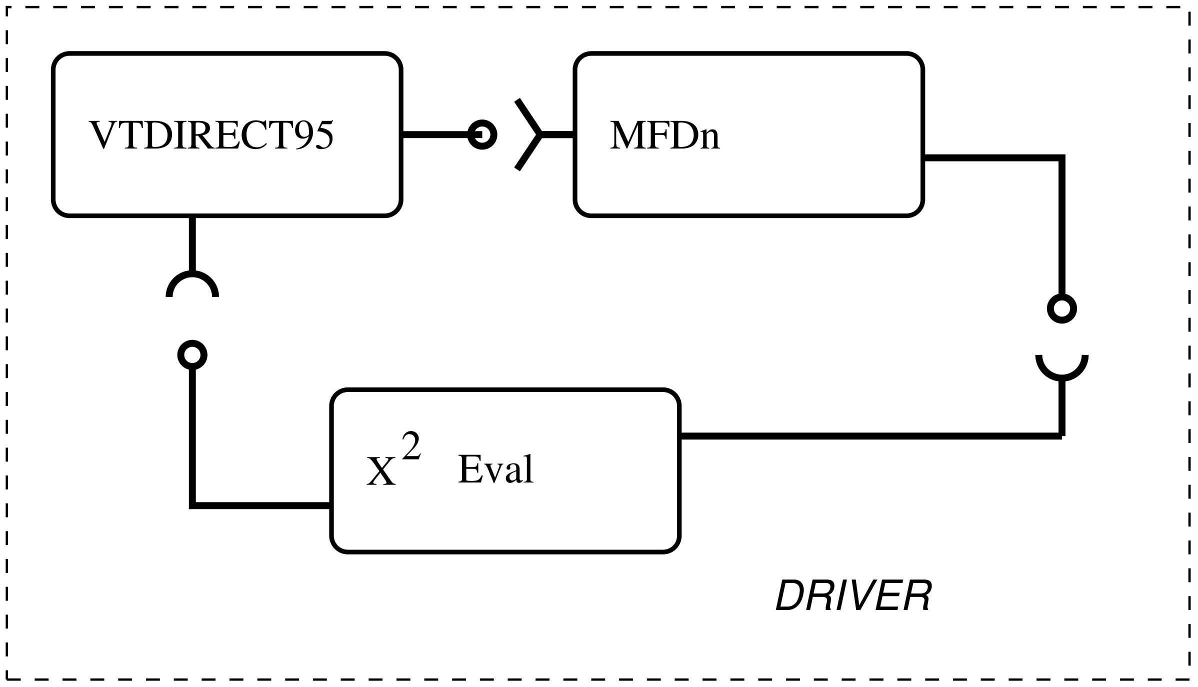

Figure 3.3 Component Level Diagram of the integration system

the optimization code VTDIRECT95, the ab-initio nuclear calculation code MFDn and the objective function evaluator χ^2. A driver is build around to synchronize the data and event flow between various components. Figure Fig. 3.3 describes the component level diagram of the integration system. The high level data flow suggest that optimization code will provide a set of input parameter (sample points) to the MFDn code, which would in turn generate the input for objective function evaluator. Objective function evaluator would calculate the CF and will feed its output to optimization code which would then generate sample points for next iteration.

3.1.2 Design of χ^2 Function

The chi-square distribution is a special case of the gamma distribution. In probability theory and statistics, the chi-square distribution (also chi-squared or χ^2 distribution) is one of the most widely used theoretical probability distributions in inferential statistics, e.g., in statistical significance tests (30) (31). It is useful because, under reasonable assumptions, easily calculated quantities can be proven to have distributions that approximate to the chi-square distribution if the null hypothesis is true. The best-known situations in which the chi-square distribution are used are the common chi-square tests for goodness of fit of an observed distribution to a theoretical one, and of the independence of two criteria of classification of qualitative data. Therefore design of χ^2 function plays a important role in how fast and with what accuracy the computed and experimental results would match.

The χ^2 used in this experimentation is constructed using a theory file, an experimental file, and the base energy value of the given nucleus. The theory file is an output from the MFDn code which contains calculated observables. The experimental file has the energy levels as found experimentally by different national scientific organizations (**?** ; **?**). In addition to the energy values, each energy level is associated with the spin j of the protons/neutrons. This spins are used to match different levels in theoretical and experimental observables. There are many options in construction of χ^2. A particular choice depends on such parameters as the quality of the experimental data and the questions nuclear physicists want to answer comparing the theoretical and experimental energy levels. As an example consider the following χ^2 definition:

$$\chi^2(\mathbf{v}) = \sum_{\substack{1 \le i_t \le 15 \\ 1 \le i_e \le k}} \left[E_{i_e}(\mathbf{v}) - \tilde{E}_{i_t}(\mathbf{v})\right]^2 \times \sigma_{i_e}^2, \tag{3.1}$$

where $\mathbf{v} = (V_0, V_1, V_{tens})$ and $\tilde{E}$ are the absolute experimental and theoretical energies, respectively; i_e and i_t are the indices of the corresponding *matched* energy levels (one i_e paired with one i_t).

Each experimental energy level l_e is assigned the weight σ_{l_e}. This weight is inversely proportional to the distance of that energy level from the ground energy level. Many nuclei have energy states where nuclei has more then one or an undecided spin. While matching those experimental energy states it is important to decide their contribution towards calculation of χ^2 function. Therefore, assignment of weight is an important step-in designing χ^2 function. The weight assignment used for the experiments done in this paper can be described mathematically as

$$\sigma_{l_e} = 1/2^{l_e} \tag{3.2}$$

3.1.3 Design consideration for Driver

It can be derived from the observations stated above that both MFDn and VTDIRECT95 have well-defined input and output interfaces. The goal is now to tunnel the sample points in search space as generated by optimization technique appropriately to MFDn, calculate the χ^2 function value for the MFDn run and tunnel back this result to optimization algorithm. Consider the following external additions (stubs) necessary to interface MFDn and VTDIRECT95:

1. Input Modifier (IM). Parameters generated by optimization technique needs to be inserted at proper places into an input file for MFDn. IM takes the set of parameters from the optimization code, their corresponding position relative to the first parameter and inserts them to the proper input file required by MFDn code.

2. Wait (W). Once the parameters are written in input file, appropriate commands are issued to spawn the MFDn run for the current evaluation. The flow of program needs to be halted until MFDn code has completed the evaluation of parameters. W is essentially responsible for halting the program until MFDn code completed it evaluation. When multiple nuclei are considered in a fitting process, W waits until it gets completion signal from all the MFDN process running different nuclei.

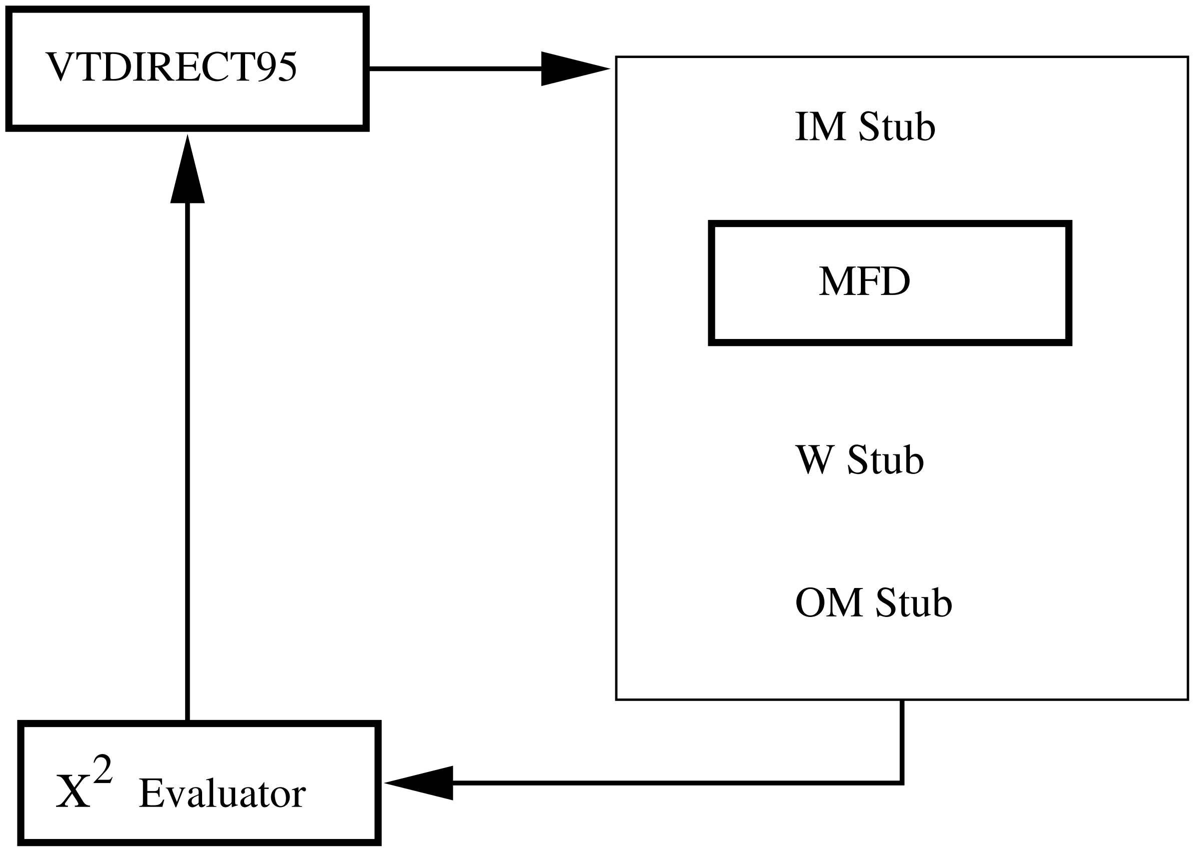

Figure 3.4 Work flow diagram for the MFDn and VTDIRECT95 supplemented with stubs

3. Output Modifier (OM). It is essentially post processing the output produced by MFDn process. After the completion of MFDn process, output files, such as theoretical observables, are generated. The output files have data such as excitation energies and their relative nuclear spins for a given nucleus for the desired number of states. This output file needs post-processing in order to become suitable input for χ^2 function calculation.

Fig. 3.4 shows a work flow between the VTDIRECT95 and MFDn codes aided by the described additions.

3.2 Implementation

While initiating a optimization search run there can be multiple choices for configurations. Two distinct and widely used configuration are as :

- Single Nucleus Multiple Parameters. In this configuration, single nucleus takes part in the fitting process for a given set of parameter.
- Multiple Nuclei Multiple Parameters. In this configuration, multiple nuclei take part in the fitting process for the same set of parameter.

Both of the above stated configurations can be run either in sequential or in parallel mode of optimization technique. Each such mode of optimization technique are same at the core level but differs slightly in the manner of how many sample points are evaluated at any given instance. Sequential VTDIRECT evaluates each sample point in sequential order, so it can be safely assumed that there is only one worker that evaluates the point. In Parallel VTDIRECT algorithm each sub-domain master is responsible for generation of points in there domain. Each such generated points are then evaluated by workers in parallel. When multiple nuclei are used in experiments for fitting the parameters, each worker is responsible for evaluating all the nuclei for a given set of parameters.

While running multiple nuclei in sequential mode, multiple instances of the same nucleus would be running, with same set of parameters in their input file. It is important to maintain unique identification for each nucleus so that concurrently running MFDn process do not write the output files to the same path, other stubs used for post processing of output files do not pick the wrong output file for processing. The situation gets a higher level of complexity when multiple nuclei are run in parallel mode, for in that situation, multiple workers would evaluate the sampled points simultaneously and each worker in turn would run instances of all the participating nuclei. It is necessary once again to maintain enough uniqueness so that no two files, in input or output, are overwritten or overlapped. This brings a couple of design consideration for each worker.

- Each worker should be able to write to separate input files for each nuclei.
- Each worker has to wait at 'WAIT' stub for completion of all the nuclei runs before it can proceed further.
- Each worker should be able to store output for different nuclei at appropriate places.

There can be a couple ways through which this problem can be addressed.

- Create different input/output filename at run time and make MFDn code flexible enough to accept different file names. In this scenario each worker would append some unique characteristic to the filename it is creating as a input to MFDn process and would expect output in a similar format from the MFDn code. This unique naming convention would then allow to identify each run uniquely.
- Setup a directory structure for calculations making use of the unique worker identification number, the evaluation number and the nuclei number. This unique characteristic can be used to create subdirectories for each worker-nuclei pair so that the MFDn runs with different nuclei, as performed by different workers, can be identified uniquely.

Changing input or output file name according to a nucleus name or a iteration number at MFDn code level defeats our efforts for seamless integration for it needs to change the MFDn code interface so that it is molded to accept the input in the way we like and similarly produce and output which can sustain this current design. Similarly doing it from VTDIRECT95 process level is not helpful either. Therefore, first approach can not be considered as a part of the solution. The second approach takes the advantage of the fact that each worker in the worker pool of VTDIRECT95, each evaluation and each participating nucleus can be uniquely numbered. As soon as a worker gets a sample point for evaluation, it creates a directory, named after its unique identifier. This directory is now the play ground for that worker. For each nucleus ,that are to be evaluated for a given sampled point, worker creates a sub-directory, named after the nucleus number respectively. In this arrangement, each worker has its own play ground, so no two workers can actually can get confuse with the set of each other's files. Within the directory structure of each worker, each nucleus has its own sub-directory and therefore a worker can not overwrite files of a nucleus into the path of other nucleus. Similarly, post-processing of the output files, that includes book-keeping process of generated observables and calculation of χ^2 would never be erred for two different nucleus.

The current setup interfaces with VTDIRECT95 and MFDn code, but it also requires some input to bridge the absence of information flow from VTDIRECT95 to MFDn code. This information includes the positions where the sample points, as generated by the search algorithm, needs to be inserted in the input file for MFDn code and names of the nuclei which would participate in the fitting process. This information is provided by the configuration file, $nuclei\_config.dat$.

3.3 Alternative Design for High Performance Clusters

Earlier suggested design and implementation for parallel integration code relies on the fact that dynamic threads can be spawned from a running process. While working on a self monitored or controlled cluster the schema mapping is quite apt, but running the integration code on High Performance Clusters pose a different run time constraints on the design. Almost all of the High Performance Computing clusters do not provide dynamic spawns from a running process. A design that suggests the possible modification in the proposed implementation is detailed below.

‘The Producer/Consumer design pattern is based on the Master/Slave pattern, and is geared towards enhanced data sharing between multiple loops running at different rates. As with the standard Master/Slave design pattern, the Producer/Consumer pattern is used to decouple processes that produce and consume data at different rates. The Producer/Consumer patterns parallel loops are broken down into two categories; those that produce data, and those that consume the data produced’ (39).

It can be observed from the earlier mentioned text that the optimization technique is responsible for generation of sample points, hence can be referred to as ‘Producer’, while MFDn code is responsible for evaluating those points or in other sense are responsible for consuming those points and therefore can be termed as ‘Consumer’. The time taken by MFDn code to make a evaluation run is linearly dependent on the Hamiltonian size of the matrix for that nucleus. Hence, time taken by MFDn code to calculate observables for different nucleus would be different. Since sample point generation from the optimization code occurs in a comparatively small time, for parallel integration code, rate of generation of point would be faster then the rate their evaluation. Working along this characteristic it can be observed that the problem statement of the thesis suits well with Producer-Consumer design pattern.

The suggested design is based concept of classic Producer-Consumer paradigm. As observed earlier the parallel optimization techniques samples multiple points and VT-DIRECT workers evaluate them in parallel. Here producers would be the workers from VTDIRECT95 pool, which on receiving a sample point would put the input file for MFDn code in their directory, refer 3.2 or in other words would buffer them. The consumers would be the MFDn calculation code, which would perform MFDn code run for each such generated sample points. Multiple instances of MFDn code for participating nuclei will start on a fixed number of processors with a distinct identification number. Each such instance of ab-initio calculation code, which is consumer in this case, would look for work from producers, which are workers from VTDIRECT95 worker pool.

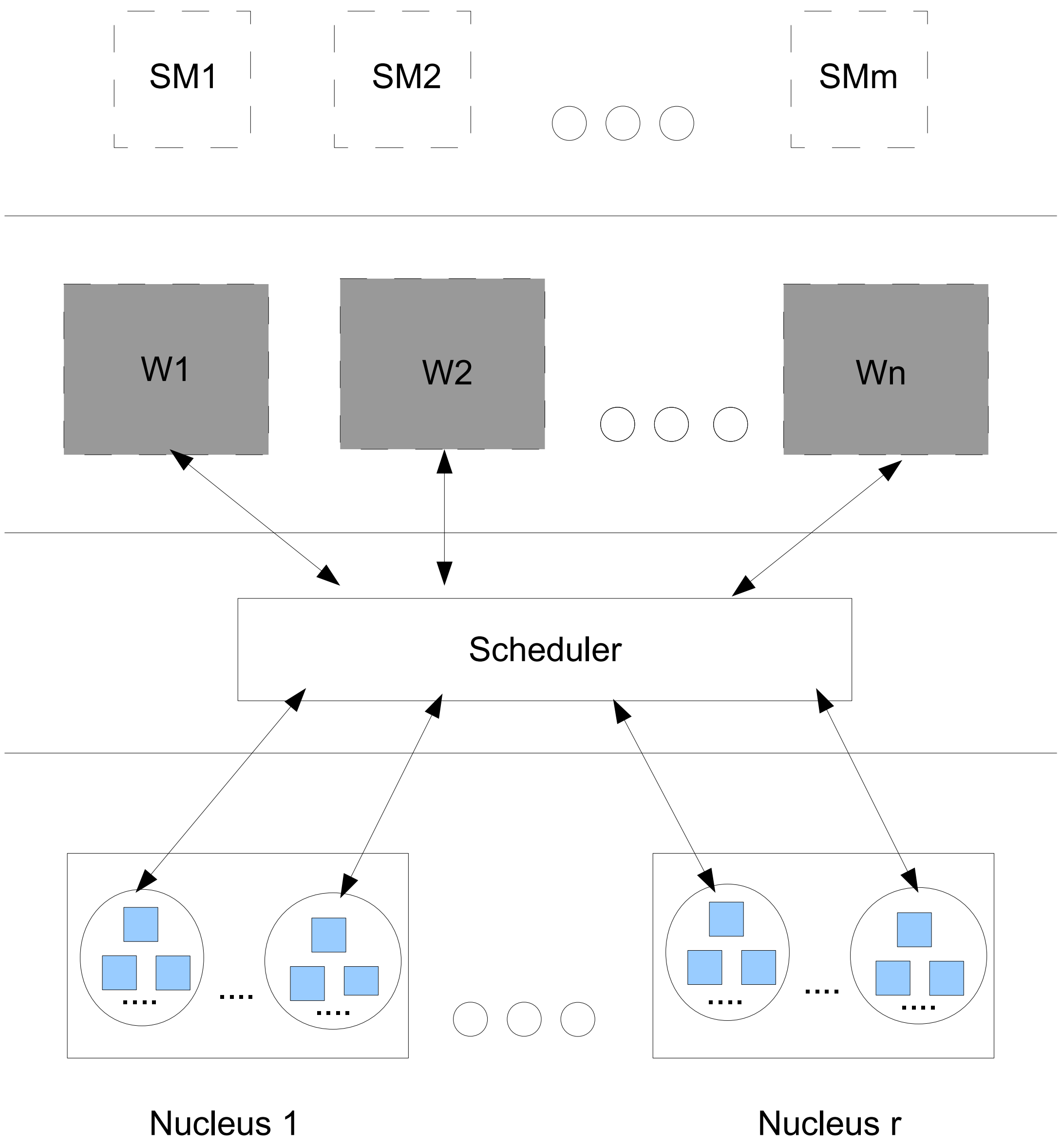

Figure 3.5 Design for implementing automation code for High Performance Cluster

Scheduler would act as a 'man in middle', which would seek any unattended work from producers (VTDIRECT workers) and then look for any idle MFDn consumer. If a consumer is idle, it will attach the work from producer to the MFDn consumer. If none of the MFDn consumer is idle, it will wait until the point in time on of the MFDn consumer is free. Thus, Scheduler would be responsible for getting the work from the producers and give them to the consumers.

Scheduler will, in general, have a data queue and a worker queue. The producers would put their work request in data queue and the workers will maintain their status in worker queue. Each nucleus would have its own data and a worker queue. One way to approach the design of the Scheduler is to have a 'Global View'. In this schema, any producer can be served by any worker, i.e.. each producer would submit its work request to the Scheduler, in data queue, which would in turn look through all the available workers, in worker queue, seeking for an idle worker. If a worker is found idle, the sample point, as generated by the producer (VTDIRECT worker), would be given to the consumer for evaluation.

3.4 Alternative design for Grid Computing

Gird computing is one of the emerging technology that has provided a new horizon in the field of resource utilization. As defined by IBM, 'Grid computing allows you to unite pools of servers, storage systems, and networks into a single large system so you can deliver the power of multiple-systems resources to a single user point for a specific purpose. To a user, data file, or an application, the system appears to be a single enormous virtual computing system' (15).

One of the critical requirement for any application to use Grid Computing is to have software that can be farmed out in pieces and can be executed on multiple computers. It can be referred that the components described during this work (VTDIRECT95, χ^2

Calculator and MFDn code) can easily be distributed amongst multiple networks as they are all loosely coupled. The suggested design to solve the problem statement of the thesis on grid computing is a minuscule step which would present basic framework considering Workload Management, Scheduler and Resource Management. Security and User Interface is left has the milestone for the next iteration.

This design assume that VTDIRECT95 would run on same cluster, that is, we do not apply grid computing for running VTDIRECT95 code. Similarly, a single MFDn nucleus run would be executed on a single cluster. This assumption is made due to the fact that, the communication and I/O overhead required by grid computing would be much more in magnitude then the gain of using unused processor at different locations. Therefore a valid scenario for the schema would be as shown in Figure Fig. 3.6:

The figure explains that VTDIRECT95 code would run at one geographical location, and the MFDn code is running at another geographical location. If running with multiple nucleus, each instance of nucleus can be distributed to different geographical location or clusters. Again, Producer/Consumer design schema is applied to the current design. Producer would be the VTDIRECT95 process which would produce sampled points for evaluation. On the same cluster as that of VTDIRECT95, say VTDIRECT site , there would be a 'Resource Manager' process. The Resource Manager process would have information like number of clusters attached to the gird, method to access those locations and number of characteristic MFDn process running on each cluster or site. The functionality of Resource Manager process can be listed as :

1. Collect the work (sample points) from Producers (VTDIRECT workers).

2. Calculate the amount of work needed to send to each site.

'Scheduler' process on the same site as that of VTDIRECT would then be responsible for the following functionality.

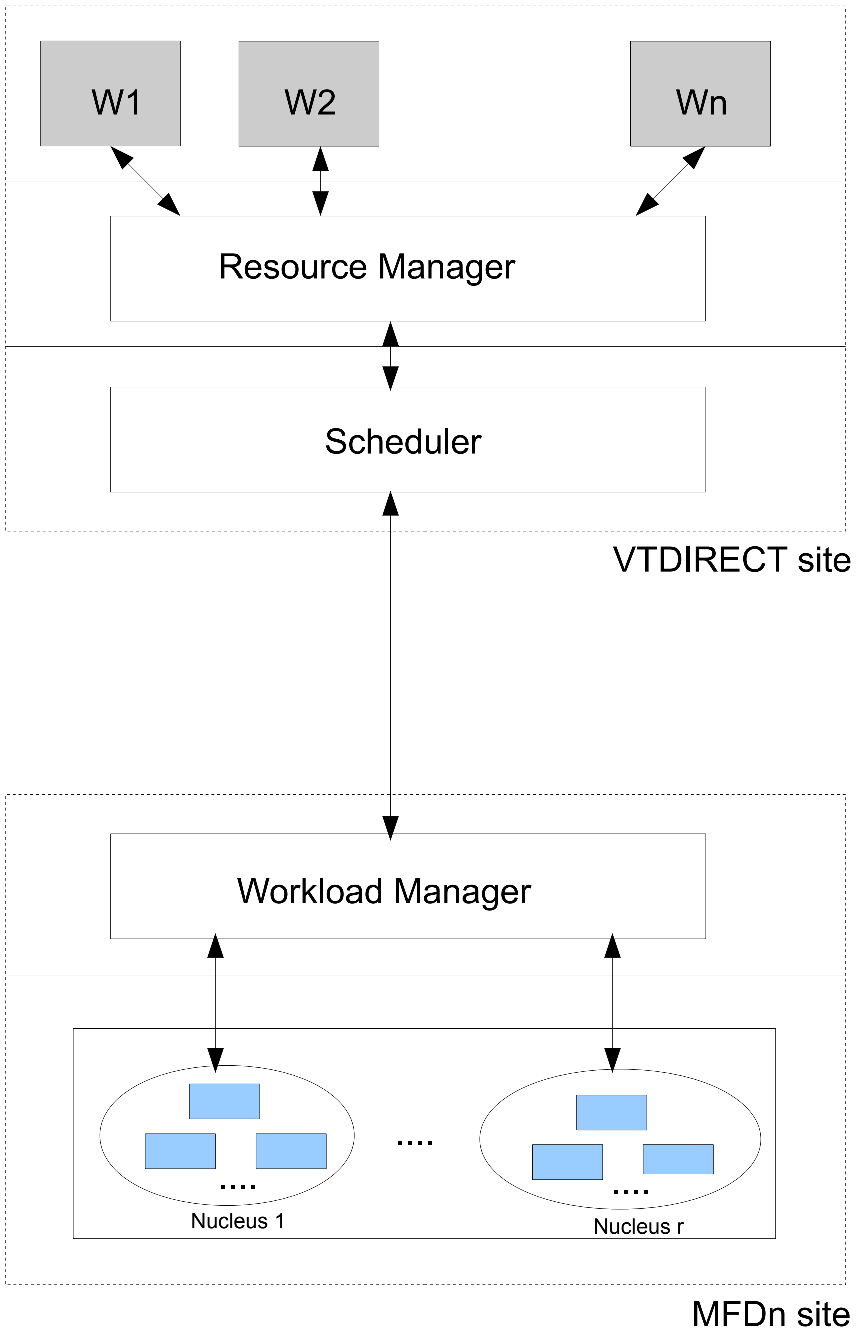

Figure 3.6 Design for implementing automation code for Grid Computing

1. Establish connection to the remote site.
2. Send the data to the appropriate site.
3. Collect data from the appropriate site and channel it to originating VTDIRECT worker.

Each such connected sites, say MFDn sites, will have a 'Workload Manager' process. This process would run on each site that would be connected to VTDIRECT site. The primary responsibility of this process would be:

1. Accept the incoming connection and data from the VTDIRECT site.
2. Schedule and distribute the work to the running MFDn process.
3. Send the calculated observables back to the VTDIRECT site.

3.5 Performance Monitor

As described earlier, running time for the ab-initio calculation is dictated by the dimensionality of the Hamiltonian matrix of a nucleus. Higher the dimensionality of the Hamiltonian matrix, larger will be the time taken for an execution as compared to the one with a Hamiltonian matrix of smaller dimensionality. While running a multiple nuclei case, each sample point needs to be evaluated by each nucleus and the final χ^2 is the sum of calculations of χ^2 of individual nucleus. When this nuclei are run in parallel, the total time of one evaluation would be dictated by the nucleus with the biggest Hamiltonian matrix. To make maximum utilization of the resources available, it is in best of interests to reduce this bottleneck. To reduce this bottleneck, processor allocation for each nucleus needs to be changed. Nuclei with lighter Hamiltonian matrix should be given smaller number of processors while the one with heavier Hamiltonian matrix should be given more number of processors to speed up these calculations.

This function is performed by 'Performance Calculator' module. During evaluation of each sample point, MFDn executable for each participating nuclei is called by a worker. This executable is forked as a child process and a timer is attached to each child thus forked. As soon as child completes its evaluation and reports back to parent process, another snapshot of the time stamp is taken. Having timestamps at start and end of child process gives us total running time of the child process. Time statistics are collected for predefined n evaluations. Gathering such information helps us decide what could be the proper allocation of processor for existing nucleus to have optimum usage of resource.

For each participating i_{th} nucleus, average runtime $local\_average\_i$ for n evaluations is calculated. Then the average times from all the nuclei are added together to form $global\_average$. Thereafter, ratio of $local\_average\_i$ over $global\_average$ is calculated as δnuc_i. Then decision for allocation of processor to each nucleus follows the rule as :

1. $\delta nuc_i \leq 1 - \epsilon$; then increase the number of processor to the next MFDn magic number.

2. $\delta nuc_i \geq 1 + \epsilon$; then decrease the number of processor to the previous MFDn magic number.

The idea behind the formulation is to bring the average running time of each nucleus within a user defined interval from each other. This interval can be set by modifying ϵ and thus tolerance time can be set accordingly.

CHAPTER 4 RESULTS

Having discussed about the proposed design, various configurations and their implementation, this section presents results from experiments and plots as to how the above contributions have met the problem statements of this thesis. The design implemented for obtaining the results for this section is 3.1. This chapter is divided into two sections. First section, details various hardware platforms used for experimentation and the basic setup of the test, including the input files and the final output files as generated by the current setup. Second section provides results from different test runs, respective graphs and their interpretation.

4.1 Platforms

An implementation of the proposed design for single and multiple nuclei has been developed and tested. It integrates the serial and the parallel VTDIRECT95 and parallel MFDn codes. Computing platform at the National Energy Research Scientific Computing Center (NERSC) at the Lawrence Berkeley National Laboratory and Computing platform at Scalable Computing Laboratory, Iowa State University served as testbeds for the development and testing of serial and parallel version of the proposed design. Specific systems that were used are as:

- Bassi. Bassi is a cluster at NERSC configured with IBM p575 POWER 5 system. It is a distributed memory computer with 888 processors (comprising 111 nodes and sharing 32 GBytes of memory) used for scientific applications. Each Bassi processor has a theoretical peak performance of 7.6 GFlops, and the nodes are interconnected by the IBM “Federation” HPS switch.

- Jacquard. Jacquard is a 712-CPU Opteron cluster running a Linux operating system. Jacquard has 356 dual-processor nodes available for scientific calculations. Each processor runs at a clock speed of 2.2GHz, and has a theoretical peak performance of 4.4 GFlop/s. Processors on each node share 6GB of memory. The nodes are interconnected with a high-speed InfiniBand network. Shared file storage is provided by a GPFS file system.

- IBM Cluster at AmesLab. IBM Cluster at Scalable Computing Laboratory, Iowa State University served as testbed for the development and testing for the parallel version of the proposed design. It consists of 22 dual-processor IBM RS/6000 43P-260 nodes with 2.5 GB RAM / node and 18 GB disk / node. The nodes are interconnected with Gigabit Ethernet network, plus Fast Ethernet.

The inputs to VTDIRECT95 are provided by the input files specified as direct**.nml file for serial implementation and pdirect**.nml file for parallel implementation. The ”‘**”’ is replaced two letter acronym of the objective function. The input file specifies number of parameters to be searched, number of iterations, number of evaluation, minimum box diameters, checkpoint flag value and some more options to tweak the performance of the optimization runs.

The current setup interfaces with VTDIRECT95 and MFDn code, but it also requires some input to bridge the absence of information flow from VTDIRECT95 to MFDn code. This information includes the positions where the sample points, as generated by the search algorithm, needs to be inserted and names of the nuclei which would participate in the fitting process. This information is provided by a configuration file $nuclei\_config.dat$.

Two sets of outputs are generated for a given run of optimization run. They are as :

- **func_value.txt** : This file contains the value of χ^2 function at each sampled point.In parallel version of the optimization algorithm, each workers writes the point it has evaluated and the function value of χ^2 at that sampled point. In serial version, as there is no parallelism, each point is written by a single process in the file. Once the files are generated, the lowest n functional valued and there corresponding sampled point can be inferred. Iteration number or sampled points associated with those functional values can there after be used to index the theory observables.

- **theory_files** : These are essentially the theory observables which are generated at each iteration. There basic requirement is for book keeping, which helps after the complete search process to look at theory files for sampled points of interest. In serial version, to uniquely identify each iteration, iteration number,ITR_NUM, is generated and the corresponding theory observables are written in a file name of type *theory_mfdns_*ITR_NUM. Each nuclei has its own sub-directory named *theory* where this observables are put. In parallel version, we can still have a common variable that can generate unique number for iteration number for all the workers, but this would imply the application of critical section and may insert some unwanted delay in the search process. So each iteration is uniquely identified by the sampled point which a worker evaluates. It is stored similarly in theory sub-directories of each nuclei as it is done for serial version.

4.2 Results

The following four nuclei have been considered for experiments : ^{47}K,^{47}Ca ,^{48}Ca, and ^{49}Ca. The Hamiltonian matrices are sparse and their sizes are 136231, 12000, 14000 and 15666 respectively, in the lowest available model space. The MFDn execution time depends heavily on the Hamiltonian size. Also, the complexity ('shape') of the objective function drives the time required by the optimization algorithm to find the minimum.

Fig. 4.1 shows the number of evaluations required per VTDIRECT95 iteration as the number of iterations grows for a sample nucleus. For a multiple nuclei fit, the runtime is guided by the runtime of the heaviest nucleus since the evaluations for all the nuclei are needed to construct the χ^2 in this case. Therefore, it is desirable to adjust the number of processors allocated to a particular nucleus based on its computational cost relative to the other nuclei in the set. In particular, for our example, ^{47}K has the largest Hamiltonian matrix, so executing it on the largest subset of processors makes sense. Fig. 4.2 depicts the case when all the three nuclei are evaluated once with three different sets of processor numbers (shown in the x-axis). The parallel time to compute all three nuclei indeed decreases as the number of processors is increased compared with the base case of using small equal number of processor for each nucleus. When about twice as many (15) processors are assigned to the calculation of each nucleus, the timings for the smaller Hamiltonians decrease by about half. The runtime for ^{47}K, however, decreases only slightly. By augmenting the number of processors to 45 for ^{47}K, the execution time is decreased dramatically, while for the smallest Hamiltonian of ^{48}Ca, the time has actually increased with the increase in the number of processors to 21. The latter indicates that the parallel overhead starts to dominate the overall execution time. In general, besides the Hamiltonian matrix size, other factors such as communication overhead and hardware characteristics, may affect the number of processors used to calculate efficiently a particular nucleus. The setup was made to run for 59 iterations and 6023 evaluations.

The serial optimization of the parameters for ^{48}Ca produced the results (Fig. 4.3) matching theoretical observables correctly to their counterparts in the experimental data file. First six states have been matched correctly with their spins and the difference in energy levels from theoretical observables and experimental data is less than 0.001 MeV. The set of parameter for the optimal result where nearly similar to those found

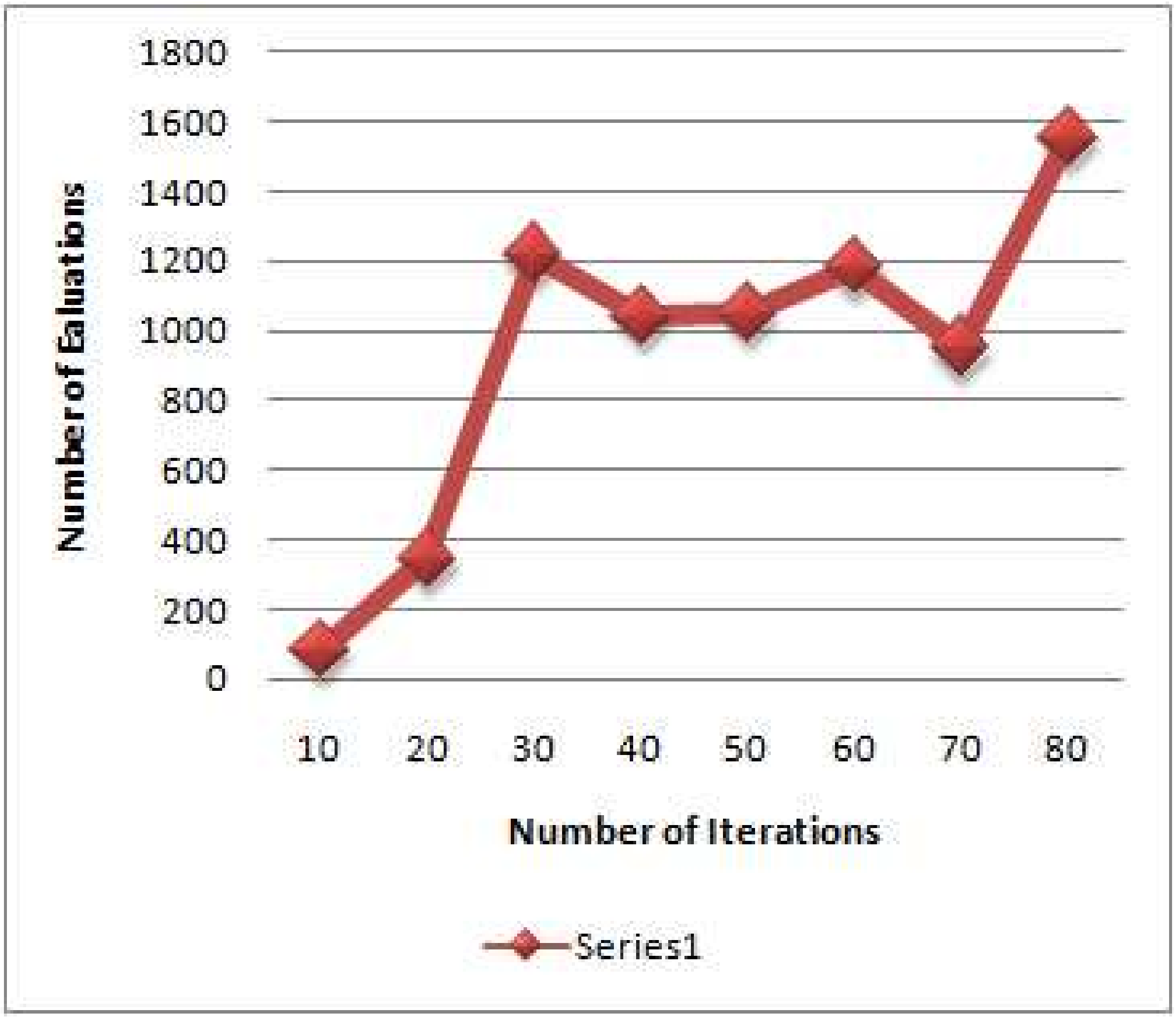

Figure 4.1 Evaluations per iteration for ^{49}Ca

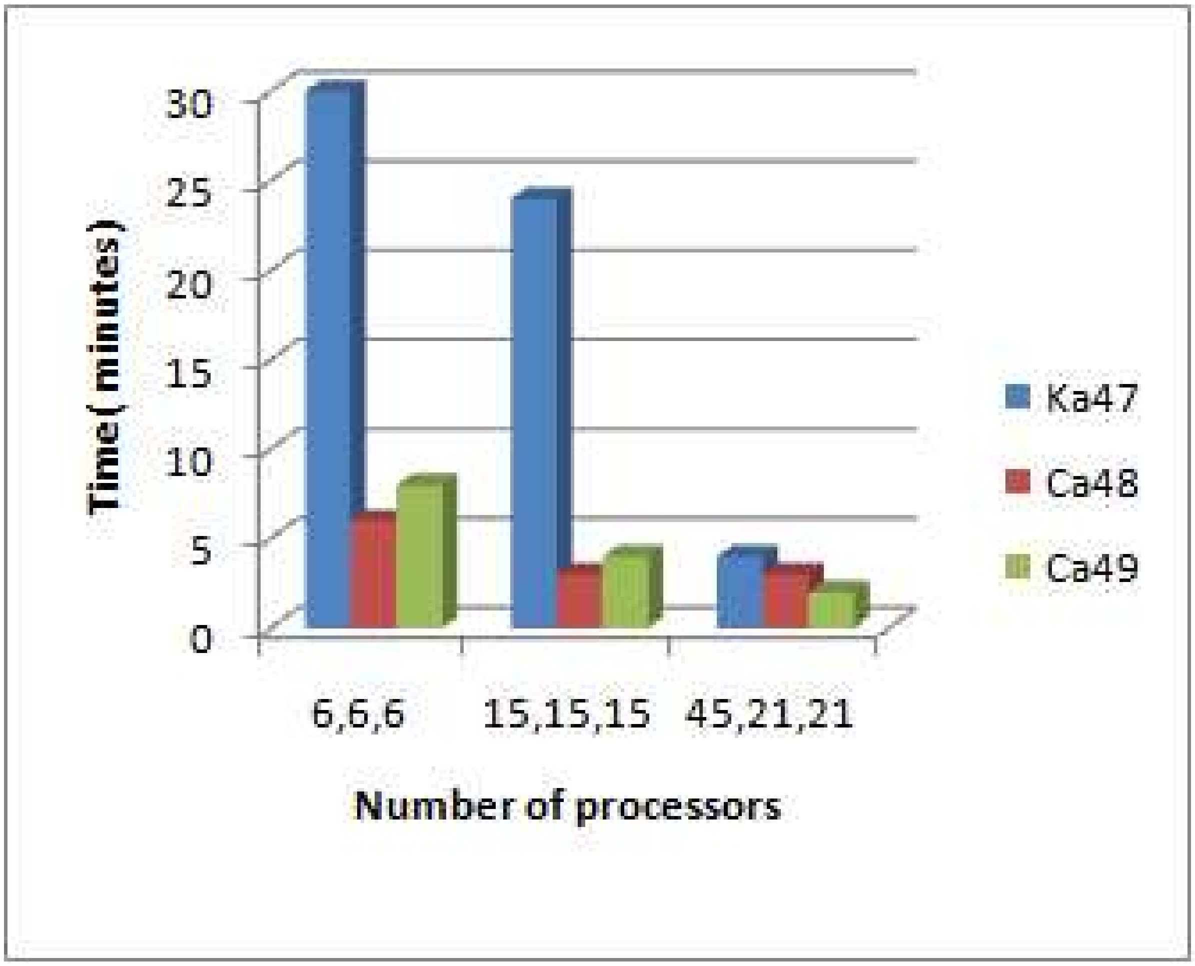

Figure 4.2 Execution times for ${}^{47}K$, ${}^{48}Ca$, and ${}^{49}Ca$ on different numbers of processors on Bassi

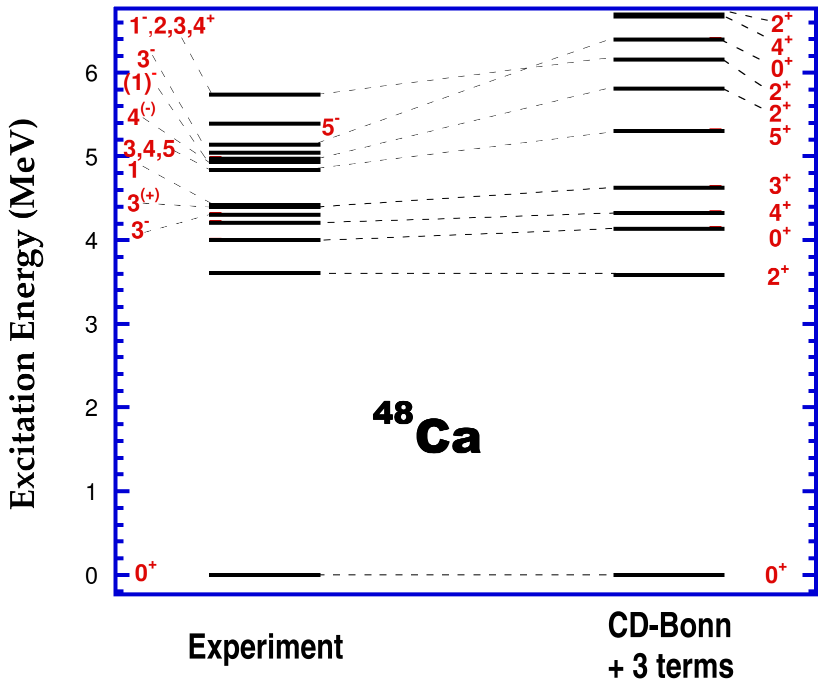

Figure 4.3 Matching of experimental and theoretical energy levels for ^{48}Ca

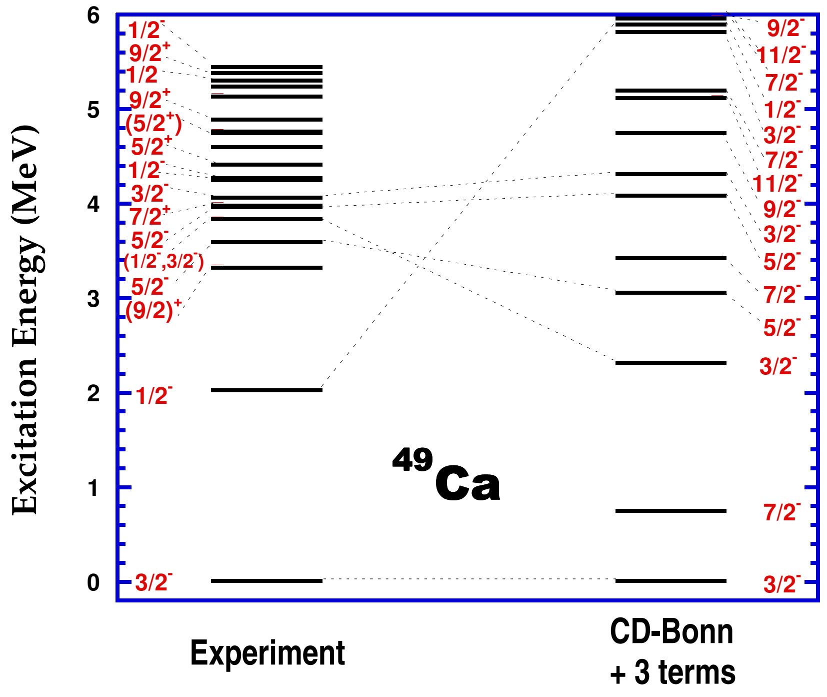

Figure 4.4 Matching of experimental and theoretical energy levels for ^{49}Ca

by physicists through manual iterations. ^{48}Ca was then run with parallel optimization technique with 2 sub-domain and 2 master configuration with optimization technique running on 4 processors. The diameter of the smallest possible potential box was set to be 10^{-3}, due to the fact that parameters considered for fitting process are not much sensitive after the third decimal place. Similar results confirming the serial search were found.

The second nucleus considered is ^{49}Ca. It was made to run with serial optimization code and the ground-state level in the theoretical observable was matched within the difference of 0.02 MeV of the experimental ground state energy value (Fig. 4.4). Higher energy levels in the theoretical observables remained partially matched to counterparts in the experimental data. A likely reason is that some of the energy levels in the experimental data have *uncertain spin levels*. In spite of this uncertainty, the first five states from the experimental data can be matched using the first ten states from the theoretical observables. Similar arguments may be used to explain the unmatched energy levels in ^{47}K.

Second run of ^{49}Ca was conducted by parallel optimization technique with 3 sub-domains and 3 masters. The optimization code was made to run on 9 processors. The parameters that were used for fitting were V_0, V_1 and V_{tens} but the domain range for V_0 and V_{tens} was modified so as to match the domain searched manually. The diameter of the smallest possible potential box was set to be 10^{-3}. Ground state energy were matched within the difference of 0.5 Mev while the ground state energy level found by manual search had a difference of $\approx$ 3.00 Mev. A better spectrum was obtained this time, refer fig Fig. 4.5, owing to fact that much more sample points were generated for the same number of iterations. The difference between excitation energies between subsequent levels in theoretical and experimental observables has decreased significantly. Still, the energy levels remains partially matched. The findings for ^{49}Ca are important from the physics point of view since they give new directions for fitting the nucleus with

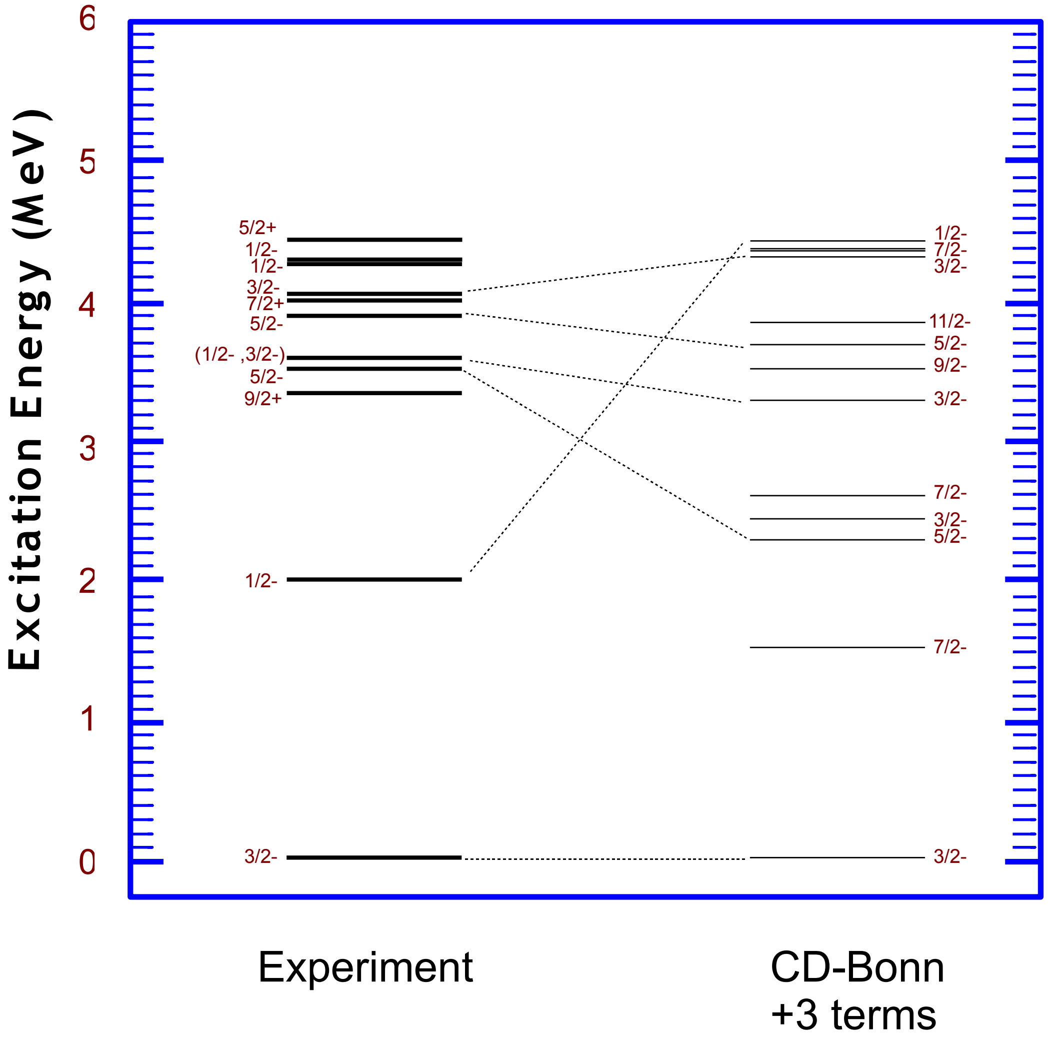

Figure 4.5 Matching of experimental and theoretical energy levels for ^{49}Ca for parallel integration code

a similar mass starting with obtained set of parameters.

The third nucleus considered is ^{47}Ca. The best result as obtained by manual search had a difference of $\approx$ 1.50 Mev. The serial optimization code for ^{47}Ca was run to fit V_0, V_1 and V_{tens}. The result obtained by the serial code is presented in Fig. 4.6. The results thus produced matched the ground energy level within the difference of $\approx$ .01 MeV. It was also made to run with parallel optimization technique with 2 sub-domains and 2 masters with optimization code running on 4 processors. It was observed that for the same number of iterations, the number of evaluation completed by parallel optimization code were more then the one completed by serial optimization code. Hence, more sample points were generated for the same number of iterations. The run provided the minimum χ^2 value obtained until now although the difference between ground energy levels of theoretical and experimental observables was increased to $\approx$ 0.3 MeV.

One of the physics goal is to fit multiple nucleus for the same set of parameters. Experiments that were conducted to have preliminary results directed towards fulfillment of this goal included ^{47}Ca, ^{48}Ca and ^{49}Ca. In this type of experiments, each sample point generated by VTDIRECT is evaluated by all the nuclei that are taking part in the fitting process. The value of objective function χ^2 for a sample point is the sum of all individual χ^2 calculated by each nucleus taking part in fitting process.

The first experiment in this series was conducted with ^{47}Ca and ^{49}Ca. The parameter that were used for fitting were V_0, V_1 and V_{tens}. The code was run with parallel optimization technique with 2 sub-domain and 2 master. The diameter of the smallest possible potential box was set to be 10^{-3}. The minimum χ^2 produced by the optimization code is very enthusiastic as the matches for ground energy level for both the nuclei where found within the approximation of 1.0 Mev. Figure Fig. 4.7 and Fig. 4.8 shows the individual matches of ^{49}Ca and ^{47}Ca found for the run. We can see that all the 10 states of ^{47}Ca as found in theoretical observables were matched within the approximation of 0.4 Mev. The ground state energy level for ^{49}Ca was matched aptly but higher states

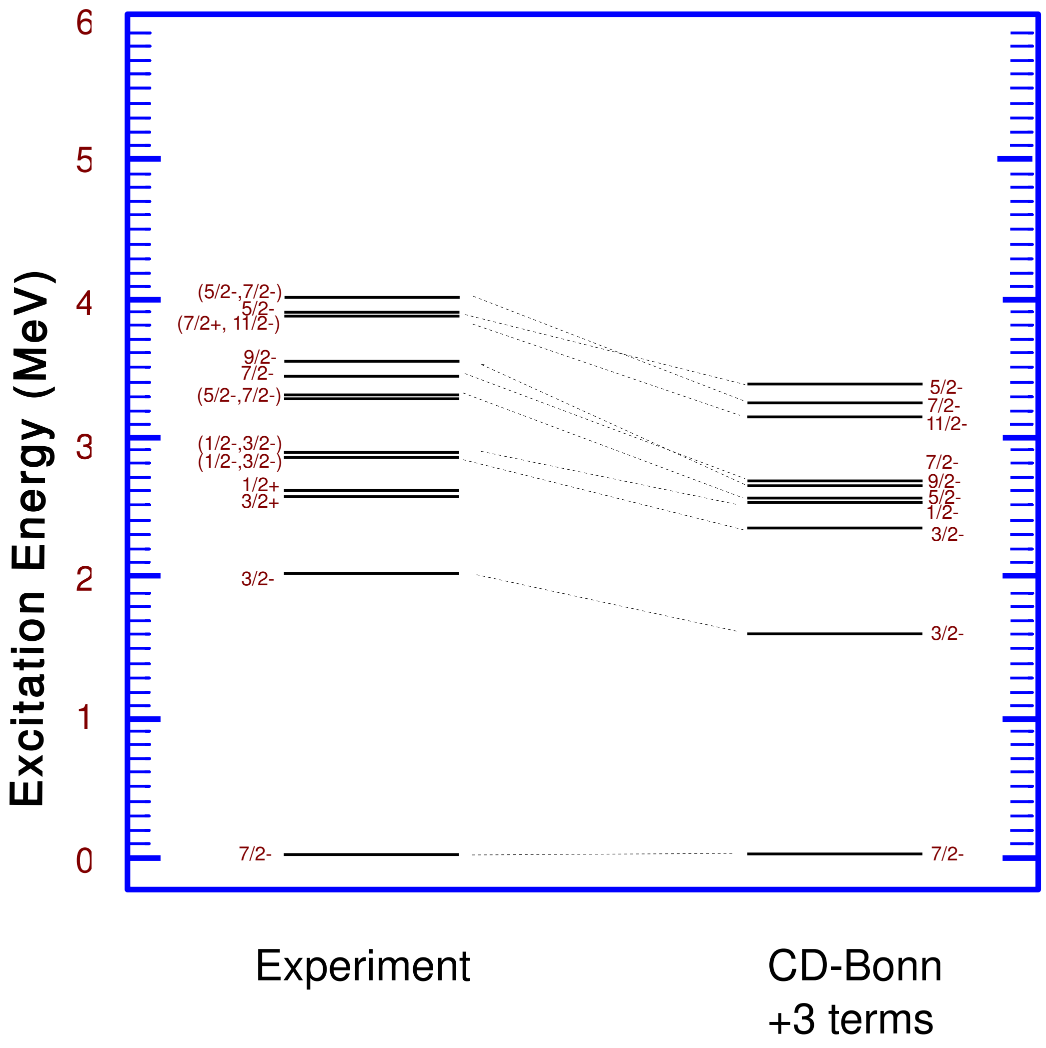

Figure 4.6 Matching of experimental and theoretical energy levels for ^{47}Ca

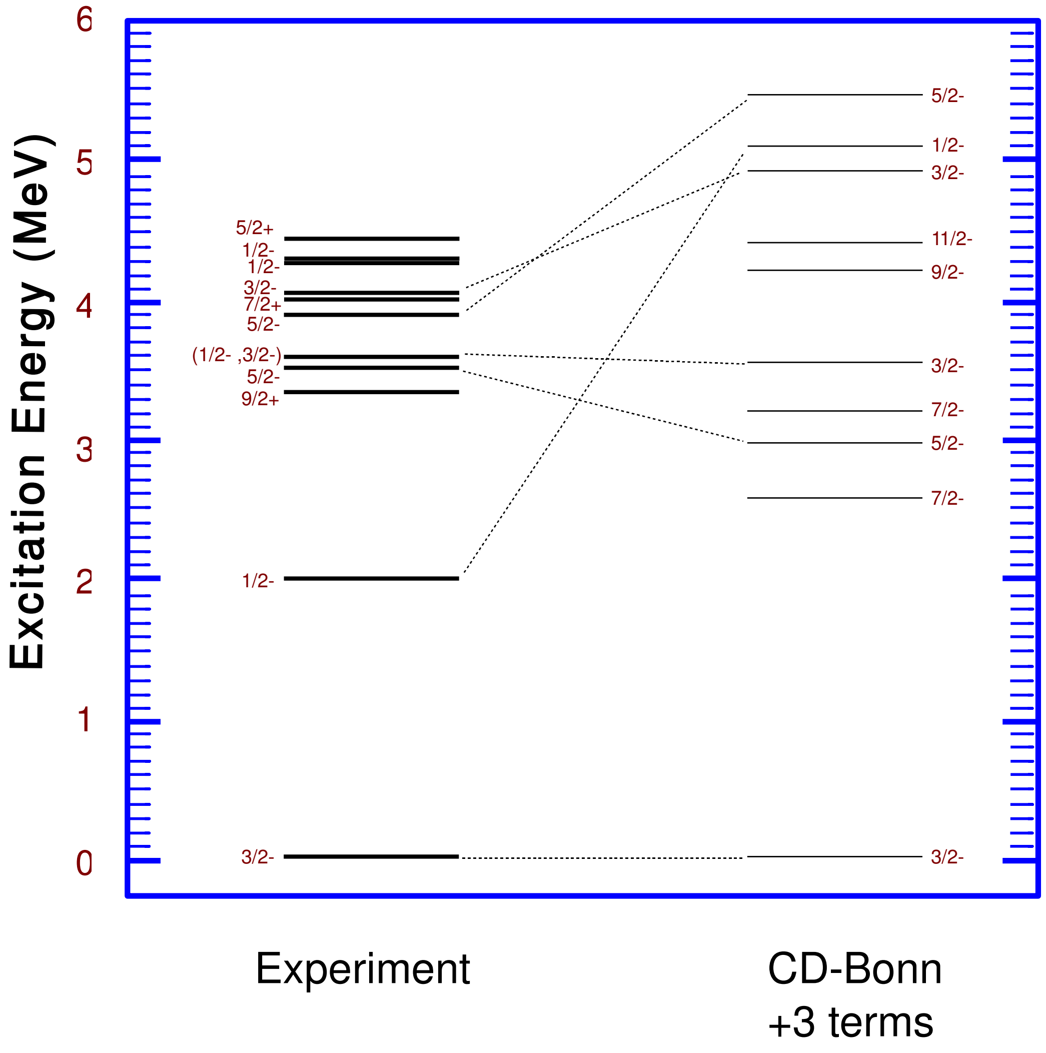

Figure 4.7 Matching of experimental and theoretical energy levels for ^{49}Ca for multiple nucleus fit

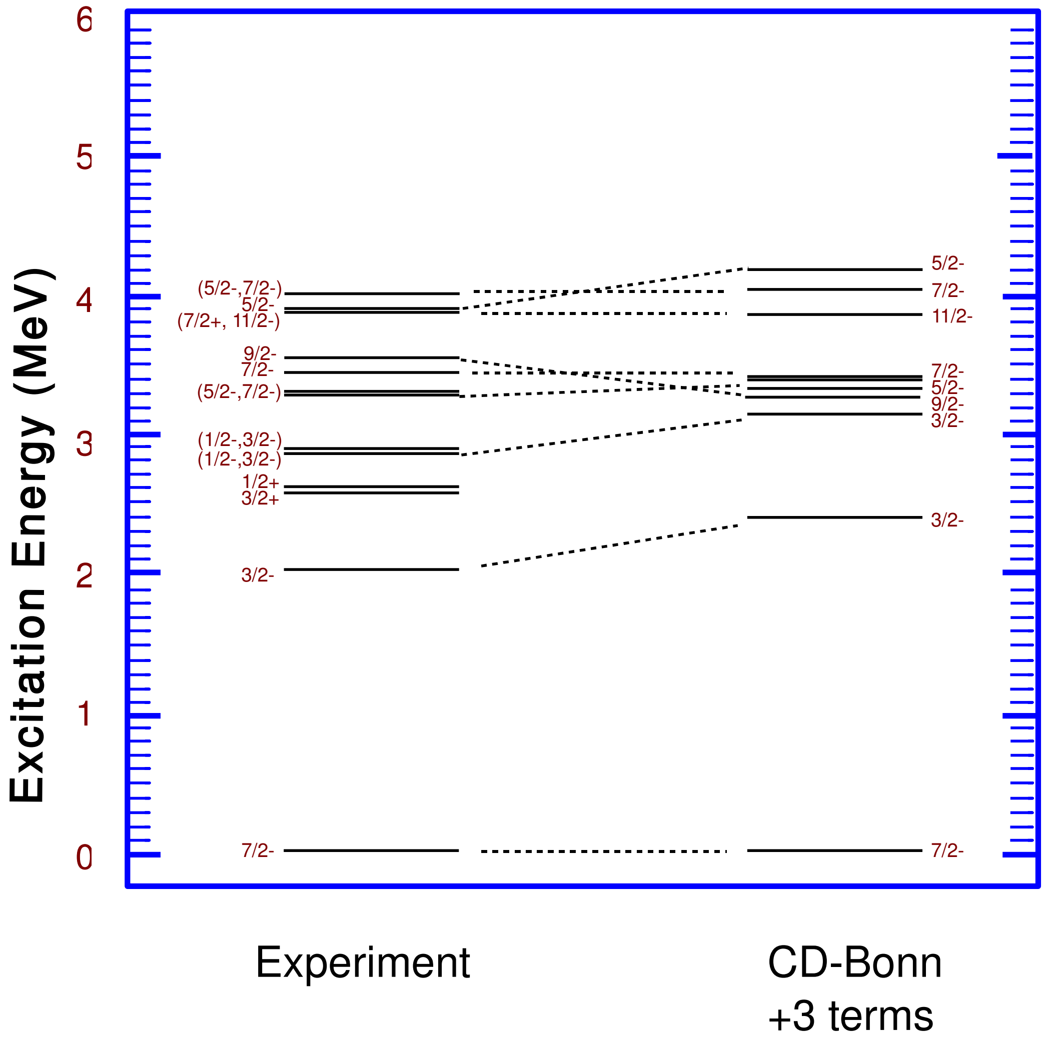

Figure 4.8 Matching of experimental and theoretical energy levels for ^{47}Ca for multiple nucleus fit

could not be matched efficiently. A likely reason is that some of the energy levels in the experimental data have *uncertain spin levels* as described earlier.

Similarly, three nuclei ^{47}Ca, ^{48}Ca and ^{49}Ca were run simultaneously to fit for V_0, V_1 and V_{tens}. Each nuclei was run on 6 processor. The optimization technique was run for around 2500 evaluations with each evaluation taking approximately 4 minutes. We have already witnessed that χ^2 is time extensive as each nuclei computation takes variable time depending upon there Hamiltonian size. So it is quite expected that this process would be more time consuming when multiple nuclei are introduced. To undertake this problem and make best uses of the resources, search was discontinued at the point where search space was seen to minimize in two different regions. Both of regions seemed promising but depending upon the individual nuclei spin and energy level, one region was chosen. The optimization method was restarted but with a restricted search space this time. The result obtained so far are not the perfect but represent the matches quite adequately.

Check pointing is a valuable feature of VTDIRECT95. We have seen an important need of this feature in the previous paragraph where the search process has to be stopped in between and then restarted again. Second need of this feature come from the fact that supercomputers with batch scheduling typically have an upper bound on the time any job is allowed to execute. For example, the maximum time permitted on NERSC supercomputers is only 48 hours, which is surely not enough to find the global or even local minimum for such an expensive function evaluation as described in this thesis. Hence, the check pointing feature is utilized as a routine procedure to restart the integrated code for the next maximum time allowed by the queuing system.

CHAPTER 5 CONCLUSIONS

5.1 Summary

This body of work has proposed a design for the integration of the MFDn and VTDIRECT95 serial and parallel codes. The automation code uses the master-worker paradigm of the VTDIRECT95 code and proposes a three-tier vertical scheme. The contribution detailed in this body of work is to show how an expensive multiprocessor function evaluation may fit into this scheme.

The body of work presented details the implementation of the proposed design for the case of a) sequential VTDIRECT95, which produces one sample point at a time and b) parallel VTDIRECT95, which produces multiple sample points at a time. Various formulation of the objective function (χ^2) were studied and the preferable choice was evaluated during the course of experiment. Using the aforementioned objective function, good matches between the theoretical and experimental energy levels were obtained for both serial and parallel code for ^{48}Ca and the ground-state energy level for ^{49}Ca. Run with multiple nuclei,^{47}Ca, ^{48}Ca and^{49}Ca were investigated and enthusiastic results were obtained that opens possibility for adding more nuclei or number of parameters to the existing search. It was also found that assigning different numbers of processors to different MFDn executions, typically in accordance with the Hamiltonian matrix size, reduces the overall time for a function evaluation needed by VTDIRECT95. Therefore, parallel integration code implements a performance monitor that monitors the running time of each nuclei and then reassigns them working processors so that over all time of single evaluation in a multiple nuclei run is optimized. In addition to the current design, an alternative design is also proposed to implement integration code for high performance cluster and grid computers.

5.2 Future Work

The implemented optimization technique, VTDIRECT95, can be replaced by other derivative free optimization technique in order to have a comparative study between optimization techniques. Similarly, objective function other then used for MFDn can be tested and experimented. The suggested design for High Performance Clusters and Grid Computers can be re-iterated and implemented which would broaden the scope of existing implementation.

Peta byte calculations are the next horizon in the filed of supercomputing. Large scale experiments can be conducted that would imbibe the essence for runs on emerging petascale platforms such as Blue Water, which would provide the computational system capable of sustained petaflop performance. Key features of Blue Water includes more then 200,000 cores, 1 petabyte memory and more then 10 petabyte of user disk storage (Blue Water).

BIBLIOGRAPHY

[1] T. G. Kolda and R. M. Lewis, and V. Torczon, Optimization by direct search: new perspectives on some classical and modern methods, SIAM Review. *SIAM Review,vol. 45, pp. 385482, 2003.*

[2] P. M. Pardalos, H. E. Romeijn, and H. Tuy, Recent developments and trends in global optimization, *Journal of Computational and Applied Mathematics,vol. 124, pp. 209228, 2000.*

[3] R. B. Schnabel, A view of the limitations, opportunities, and challenges in parallel nonlinear optimization,. *Parallel Computing, vol. 21, pp. 875905, 1995.*

[4] A. Migdalas, G. Toraldo, and V. Kumar, Nonlinear optimization and parallel computing,. *Parallel Computing,vol. 29, pp. 375391, 2003.*

[5] U. M. Garcia-Palomares and J. F. Rodriguez, New sequential and parallel derivative-free algorithms for unconstrained minimization,. *SIAM Journal on Optimization, vol. 13, pp. 7996, 2002.*

[6] R. Fletcher, Practical method of Optimization. 2nd edition, John Wiley and Sons Ltd., Chichester, UK, 1987.

[7] M.J. Box, A New Method of Constraint Optimization and a Comparison with Other Methods. Computer J. 8(1965):42-52.

[8] Herbert Spencer. The Principles of Biology, volume 1. London and Edinburgh:Williams and Norgate, first edition, 1864 and 1867.

[9] http://en.wikipedia.org/wiki/Divide and conquer algorithm [accessed 2007-07-09]

[10] D.R.Jones, Pertunen, C.D.and and Stuckman. Lipschitzian optimization without the Lipschitz constant. *J. Optimization Theory and Applications 79, 157181.*

[11] Jian He, Layne T. Watson, Masha Sosonkina. Algorithm XXX: VTDIRECT95: Serial and Parallel Codes for the Global Optimization Algorithm DIRECT *Association for Computing Machinery, Inc.*

[12] Masha Sosonkina, Anurag Sharda, Alina Negoita, James P. Vary. Integration of Ab Initio Nuclear Physics Calculations with Optimization Techniques. *Intl Conference on Computational Science (ICCS08), Krakow, Poland, June 23-25, 2008.*

[13] R. Horst, P.M. Pardalos,and N.V. Thoai. 2000. Introduction to Global Optimization.Kluwer, Boston.

[14] R. Horst and H. Tuy 1996. Global Optimization: Deterministic Approaches. Springer- Verlag, Berlin.

[15] IBM Develpoers. http://www.ibm.com/developerworks/grid/newto/

[16] Mark P. Wachowiak. High Performance and Parallel Optimization. Department of Computer Science and Mathematics, Nipissing University.

[17] C.A. Baker, L.T. Watson, B. Grossman,R.T. Haftka and W.H. Mason. Parallel global aircraft configuration design space exploration. *High Performance Computing Symposium 2000, A. Tentner (Ed.), Soc. for Computer Simulation Internat, San Diego, CA, 101106.*

[18] R.G. Carter, J.M. Gablonsky, A. Patrick, C.T. Kelly, and O.J. Eslinger. Algorithms for noisy problems in gas transmission pipeline optimization. *Optimization and engineering 2, 139157.*

[19] M.C. Bartholomew-Biggs, S.C. Parkhurst and S.P. Wilson. Global optimization approaches to an aircraft routing problem. *EUR J. Operational Research 146, 417431.*

[20] H. Zhu and D.D. Bogy. DIRECT algorithm and its application to slider air-bearing surface optimization. *IEEE Transactions on Magnetics 38, 21682170.*

[21] J. He, A. Verstak, L.T. Watson, C.A. Stinson, N. Ramakrishnan, C.A. Shaffer, T.S. Rappaport, C.R. Anderson, K. Bae, J. Jiang, and W.H. Tranter. 2004. Globally optimal transmitter placement for indoor wireless communication systems. *IEEE Transactions on Wireless Communications 3, 19061911.*

[22] K. Ljungberg, S. Holmgren and Carlborg. 2004. Simultaneous search for multiple QTL using the global optimization algorithm DIRECT. *Bioinformatics (Oxford, England) 20,18871895.*

[23] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom 2002), pp. 12-23, ACM, Atlanta, GA, September 2002.*

[24] J.W. Zwolak, J.J. Tyson, and L.T. Watson. Globally optimised parameters for a model of mitotic control in frog egg extracts. *IEE Systems Biology 152, 8192.*

[25] T.D. Panning, L.T. Watson, N.A. Allen, K.C. Chen, C.A. Shaffer and J.J. Tyson. Deterministic global parameter estimation for a model of the budding yeast cell cycle.

[26] J.P. Vary and D.C. Zheng, ibid., (unpublished), The Many-Fermion Dynamics Shell-Model Code. *Iowa State University,1992*

[27] P. Navratil, J.P. Vary, B.R. Barrett. Properties of ^{12}C in the ab-initio Nuclear Shell Model *Phys. Rev. Lett., volume 84, pages 5728*

[28] P. Navratil, J.P. Vary, B.R. Barrett. Large-basis ab-initio No-core Shell Model and its application to ^{12}C *Phys. Review., volume C 62, pages 54311*

[29] Thomas Weise. Global Optimization Algorithms - Theory and Applications. *Version: 2008-09-24. http://www.it-weise.de/*

[30] N.L. Jonhson, S. Kotz, , N. Balakrishnan. Continuous Univariate Distributions. *Second Ed., Vol. 1, Chapter 18.John Willey and Sons. ISBN 0-471-58495-9.*

[31] Alexander Mood, Franklin A. Graybill, Duane C. Boes. Introduction to the Theory of Statistics *Third Edition, p. 241-246. McGraw-Hill. ISBN 0-07-042864-6.*

[Electronic Data] Electronic Version of Nuclear Data Sheets. *telnet://bnlnd2.dne.bnl.gov*

[Blue Water] Blue Waters. Sustained Petascale Computing. *http://www.ncsa.uiuc.edu/BlueWaters/*

[32] T.W. Burrows. Nuclear Data Sheets, 74, 1; Nuclear Data Sheets 76, 191

[33] D.R. Jones and C.D. Pertunen and B.E. Stuckman. Lipschtzian optimization without the Lipscitz constant. *J. Optimization Theory and Applications, Volume 79, Pages 157-181*

[34] D.R. Jones. The DIRECT global optimization algorithm. *Encyclopedia of Optimization,Kluwer Academic Publishers, Pages 431-440*

[35] R. Machleidt and F. Sammarruca and Y. Song. Nonlocal nature of the nuclear force and its impact on nuclear structure. *Phys. Rev. C53; Phys. Rev. C63, 024001, Ref 9*

[36] J.P. Vary, S. Popescu, S. Stocia and P. Navratil. No Core Shell Model A=47 and A=49. *nucl-th/0607041/*

[37] S. Benson and L. Curfman McInnes and J.Moré and T. Munson and J. Sarich. TAO User Manual (Revision 1.9) *Mathematics and Computer Science Division, Argonne National Laboratory, ANL/MCS-TM-242, http://www.mcs.anl.gov/tao*

[38] Denitza T. Krasteva, Layne T. Watson, Chuck Baker, Bernard Grossman, William H. Mason, Raphael T. Haftka: Distributed control parallelism in multidisciplinary aircraft design. *Concurrency - Practice and Experience 11(8): 435-459*

[39] National Instruments. http://zone.ni.com/devzone/cda/tut/p/id/3023