GEOMETRY GUIDED PHASE TRANSITION PATHWAY AND STABLE STRUCTURE SEARCH FOR CRYSTALS

A Thesis
Presented to
The Academic Faculty

By

Edin Crnkic

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in the School of Mechanical Engineering

Georgia Institute of Technology

May 2012

GEOMETRY GUIDED PHASE TRANSITION PATHWAY AND STABLE STRUCTURE SEARCH FOR CRYSTALS

Approved by:

Dr. Yan Wang, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. David Rosen
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Suresh Sitaraman
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: May 17, 2012

ACKNOWLEDGEMENTS

First and foremost I would like to acknowledge and thank my advisor, Dr. Yan Wang, for his continuous support, advice, and patience throughout the past three years of research and coursework. His guidance and belief in my efforts has made the successful completion of this thesis and M.S. degree possible. Dr. Wang's technical and practical advice has proven to be indispensable.

Secondly, I am very grateful to my student colleagues, particularly Lijuan He, for her help with various technical aspects that are now part of this thesis. I would also like to thank Dr. David Rosen and Dr. Suresh Sitaraman for serving on my thesis reading committee. Much of the material I learned as part of their courses was directly applicable to this work, and I appreciate their participation in my M.S. degree work.

I also want to express my gratitude to Dr. K. Ravindra of Saint Louis University, who encouraged me to attend Georgia Tech during my undergraduate studies; more generally, I would like to thank my parents and my sister, and particularly my father, for their unfaltering belief in my abilities and their endless support of my endeavors. My academic success would not have been possible without the motivation I draw from their encouragement.

Finally, I would like to acknowledge the support of the National Science Foundation grant CMMI-1001040, which provided me with financial support throughout my graduate studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

PS	Periodic Surface
PES	Potential Energy Surface
MEP	Minimum Energy Path
NEB	Nudged Elastic Band
ROI	Region of Interest
PSO	Particle Swarm Optimization
RG-PSO	Random Group Particle Swarm Optimization
AG-PSO	Active Group Particle Swarm Optimization
QE	Quantum Espresso
BFGS	Broyden-Fletcher-Goldfarb-Shanno
FeTi	Iron Titanium
VO_2	Vanadium Dioxide
FePt	Iron Platinum

SUMMARY

Recently a periodic surface model was developed to assist geometric construction in computer-aided nano-design. This implicit surface model helps create super-porous nano structures parametrically and support crystal packing. In this thesis, a new approach for pathway search in phase transition simulation of crystal structures is proposed. The approach relies on the interpolation of periodic loci surface models. Respective periodic plane models are reconstructed from the positions of individual atoms at the initial and final states, and surface correspondence is found using a Simulated Annealing-like algorithm. With geometric constraints imposed based on physical and chemical properties of crystals, two surface interpolation methods are used to approximate the intermediate atom positions on the transition pathway in the full search of the minimum energy path. This hybrid approach integrates geometry information in configuration space and physics information to allow for efficient transition pathway search. The methods are demonstrated by examples of FeTi, VO_2, and FePt. Additionally, two new particle swarm optimization (PSO) algorithms are developed and applied to crystal structure relaxation of the initial and final states. The PSO algorithms are integrated into the Quantum-Espresso open-source software package and tested against the default Broyden-Fletcher-Goldfarb-Shanno relaxation method.

1. INTRODUCTION

This chapter presents an overview of the material that will be discussed throughout the thesis. In section 1.1 the problem to be studied is discussed in a broad sense, and section 1.2 explains the importance of the problem in terms of real life application. Section 1.3 discusses in more detail the approach that will be taken to solve the problem. Finally, section 1.4 outlines the novel contributions that are made as part of the work in this thesis.

1.1 Problem To Be Solved

Properties of materials are of great interest to scientists and engineers; before a material can be applied, its characteristics must be well understood. The choice of material can be the difference between a successful design and complete failure. As a result, the study of materials has long been important to researchers. Traditionally, this involves laboratories, material samples, and dedicated machinery for experimentation and testing of mathematical predictions. Since the advent of the computer and computer modeling, however, an alternative approach to the study of material properties has emerged. Although the traditional laboratory settings are still common, more and more researchers have focused on computer aided modeling of materials and material behavior under various conditions. Techniques like the finite element method are being used heavily to predict and visualize behavior. There has also been a focus on material modeling on micro- and nano-scales, which can be used to gain a deeper understanding of material behavior as a result of loading and phase transitions.

However, modeling and simulation at these small scales have proven to be quite difficult. In this thesis, the focus is on nano-scale modeling of crystal structures, as well as the simulation of the structures' transition between different phases. With the observation that hyperbolic surfaces exist in nature ubiquitously and periodic features are common in condensed materials, recently an implicit surface modeling approach was proposed, the periodic surface (PS) model, which can represent geometric structures at nano scales. Periodic surfaces are applied as either loci or foci in geometry construction. Loci surfaces are fictional continuous surfaces that pass through discrete particles in 3-dimensional space, whereas foci surfaces can be regarded as isosurfaces of potential or density in which discrete particles are enclosed. This surface model allows for parametric construction of highly porous structures from atomic scale to meso scale. The PS model can be used to simplify the task of modeling the material, as well as provide a way to simulate phase transitions. In combination with previously developed methods, this gives us a method for prediction of activation energies and material behavior in nano- to meso-scales.

When simulating the transition of a material from one phase to another, we start with known initial and final states. The arrangement of the structure in these two states is found experimentally and available in literature. The purpose of the phase transition simulation is to find the activation energy, or the amount of energy that must be transferred into the system to complete the transition from initial to final state. Total potential energy of the system can be calculated at the initial, final, as well as all intermediate states. A set of intermediate states where the potential energy at each state is minimized is known as the minimum energy path (MEP), and the position along the MEP that has the highest potential energy is referred to as the saddle point. The difference in energy between the initial state and the saddle point is the activation energy of the transition. The search for the MEP, and the associated saddle point and activation energy, is the primary focus of this work. Current searching methods are investigated, and improvements are implemented at several steps.

1.2 Why Is This Problem Important?

As mentioned, foci surfaces can be used to model structural change in phase transitions. A phase transition is a geometric and topological transformation process of materials from one phase to another, each of which has a unique and homogeneous physical property. Understanding and controlling phase transitions is critical in designing various functional materials, such as for information storage (e.g. magnetic disk, phase-change memory, CD-ROM) and energy storage (e.g. battery, shape-memory alloy, solid-state materials for hydrogen storage). More generally, a tool which allows engineers to visualize and gather information about phase transitions would be an asset for the application of the functional materials as part of a solution to engineering problems. The creation of such a tool is the motivation behind the research presented in this work.

1.3 Problem Solution Approach

For the more limited scope of this thesis, the open-source software package Quantum Espresso (QE) is used to investigate and improve the phase transition simulation process using periodic surface models and unconstrained optimization techniques. QE is a suite of computer codes for nanoscale electronic structure calculations and materials modeling. It uses well known optimization and phase transition techniques to simulate the transition process and determine the required activation energy. The intent is to decrease computation time and improve accuracy within QE by applying the PS model and various particle swarm optimization (PSO) algorithms.

More specificially, the purpose of this thesis is to present a new geometry-guided approach to provide good initial guesses of transition path for crystals aided by the PS models so that the model construction for design and pathway search for first-principles simulation can be effectively integrated. Good initial guesses reduce the risk of being trapped in the paths of saddle points with local minimum energy. Thus the accuracy of the true pathway prediction can be improved. Additionally, various PSO algorithms are tested in place of the standard BFGS optimization algorithm for structure relaxation in QE, and the effects on simulation accuracy and completion time are investigated.

In the metamorphosis approach presented here, atom locations at the initial and final states are found in literature, and a global optimization scheme is used to move the atoms into "relaxed" positions that represent a global minimum on the potential energy surface (PES). The native BFGS optimization can be replaced with a PSO algorithm. Then PS models of the start and end crystal structures are built based on loci surfaces. Loci surface construction is used because intersections of loci surfaces present a convenient method of defining atom locations in a crystal structure. The initial guess of the transition path is represented as the interpolation between the start and end PS models in the parameter space. A method of finding the correspondence between atoms in the initial and final states will be presented and used to construct PS models. Two methods of PS model interpolation are also developed. Fig. 1 shows an outline of the foreseen computer-aided transition pathway design process. It is hoped that the closed-loop process can iteratively find a good design of materials structure and composition with the desirable transition rate. The shaded boxes show the new method presented in the thesis.

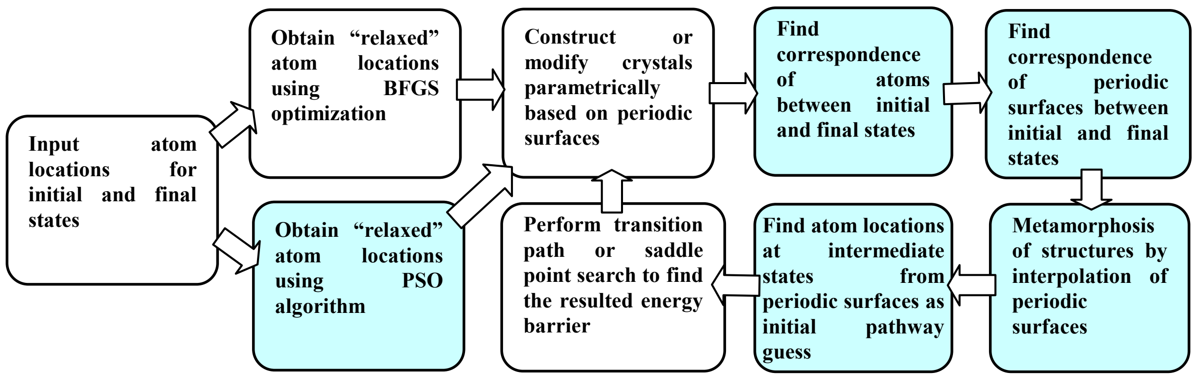

Input atom locations for initial and final states
Obtain "relaxed" atom locations using BFGS optimization
Obtain "relaxed" atom locations using PSO algorithm
Construct or modify crystals parametrically based on periodic surfaces
Find correspondence of atoms between initial and final states
Find correspondence of periodic surfaces between initial and final states
Metamorphosis of structures by interpolation of periodic surfaces
Find atom locations at intermediate states from periodic surfaces as initial pathway guess
Perform transition path or saddle point search to find the resulted energy barrier

Fig. 1: Steps of computer-aided transition pathway design

The native process in QE and the standard Nudged Elastic Band (NEB) method omits the steps seen in the shaded boxes in Fig. 1. The input consists of atom positions for the initial and final states, optimized using BFGS, and the method relies on linear interpolation of atom positions to find an initial guess of the minimum energy path. This initial guess is then used to search for the saddle point and find the energy barrier for the phase transition. Depending on the accuracy of the initial guess, the MEP that is found may have unwanted kinks or there may not be an image that captures the saddle point. As a result, the predicted activation energy may be inaccurate, and computation time depends on the accuracy of the initial guess. The intent with the additional and alternate steps shown in the shaded boxes in Fig. 1 is to improve the accuracy of the energy barrier prediction, as outlined in the following section.

1.4 Novel Contributions

The primary contribution of this thesis is an improvement of the saddle point search process using the steps outlined in the shaded boxes in Fig. 1. Rather than relying on an initial guess based on linear interpolation of positions, a method is presented to generate a more accurate initial guess to improve accuracy and computation time of the NEB method. A more accurate initial guess of the MEP reduces the aforementioned kinks, improves computation time, and helps avoid areas of local minimum energy. To this extent, images are built of the initial and final states using the PS model. Then, the correspondence of atoms between initial and final states is determined under the assumption that each atom moves to the nearest available position. A Simulated Annealing-like algorithm is used to find the correspondence of the periodic planes whose intersections represent atom positions. Then, two techniques are presented to show the metamorphosis of the structure by an interpolation of the periodic planes between initial and final states. At intermediate states, the location of atoms is again found by the intersections of periodic surfaces, and this information about the locations of atoms in intermediate states provides a more accurate initial guess of the MEP. As shown in the demonstrations, the improved initial guess leads to more accurate estimates of the activation energy compared to the empirical default initial guess used in the standard NEB method. The proposed method can be used in place of the standard NEB method for any structure composed of one or more of the 14 bravais lattices.

Before the transition simulation process can begin, the crystal structures in the initial and final states must undergo relaxation. During this process, the atom locations are optimized and moved into regions where total energy is minimized. The default relaxation method within QE relies on a BFGS global optimization algorithm to find minimum energy atom locations. As an alternative, a basic PSO algorithm is substituted for the BFGS scheme and integrated as part of the QE software. PSO is selected because it is a global optimization algorithm and should be able to replace BFGS seamlessly. It also uses multiple individuals as part of the “swarm” and investigates different regions of the search space simultaneously. Additionally, two modified versions of the standard PSO algorithm are proposed and tested. These modified versions of PSO are shown to have advantages over the original algorithm for some functions. The simultaneous search of different regions, especially within the modified versions of PSO, can more quickly find the local minima to which the individual atoms might move and therefore make the relaxation process more efficient.

In the remainder of this thesis, existing methods and concepts in geometric modeling, transition pathway search, and global optimization are summarized in Chapter 2 as a background to the proposed methods. Chapter 3 provides an overview of the geometry-guided transition pathway search and describes the method of feature-based crystal construction. It also details the morphing of surfaces between states and discusses the searching of surface correspondence based on a minimum energy change approach as well as constraints that can be applied.Finally, three example structures are tested and the results are compared to the native method available within QE. Chapter 4 discusses the original as well as modified versions of the PSO algorithm and provides data about each algorithm’s performance with standard test functions. It also details the integration of the PSO algorithms with QE. Finally, chapter 5 summarizes the findings and draws conclusions.

2. LITERATURE REVIEW

In this chapter, relevant work is presented that has already been done concerning the problems addressed in this thesis. Geometric modeling, transition path search, and saddle point search are covered in sections 2.1, 2.2, and 2.3, respectively, while section 2.4 talks about some existing ideas and methodology with particle swarm optimization.

Traditionally, phase transition is described from a top-down viewpoint as the transformation of a thermodynamic system from one phase to another. A phase is a state where all physical properties are uniform throughout the material, and the system has a particular level of free energy. When external conditions are altered, such as a change in temperature or pressure, one or more properties of the material change and a phase transition occurs. The system moves from one free energy level to another as a result of these external influences. The external conditions and amount of energy input required are quantitative measures that are used to define the phase transition. It is not necessary for the material to undergo a change in its state of matter, for example from liquid to solid. Material properties can change while remaining in the same state throughout the transition. Although phase changes in materials are a widely studied subject and much is known about it, a complete understanding of phase transitions is not yet available; even the classifications of first-, second-, and infinite- order [1,2] cannot be made without ambiguity.

Phase transition describes a wide variety of processes in diverse domains, such as liquid, ferromagnetic, superconducting, and others. In this thesis, I take a bottom-up viewpoint and refer to phase changes as geometric and topological reconfiguration, rather than the top-down classical thermodynamic viewpoint. With this approach, we are interested in changes in the material structure on an atomic scale. Structural changes in phase transitions have been found more common than previously thought. For instance, ferromagnetic phase transition was recently found to be related to crystal shape changes [3,4]. The modeling of materials and phase changes from this bottom-up viewpoint have been discussed frequently in literature (e.g. [5,6]).

The most important step involved in modeling phase transition is the knowledge of the activation energy barrier during the transition, which can be found by traversing the transition pathway. A number of methods already exist to search transition paths and saddle points on a potential energy surface (PES), where configurations with local minimal energy correspond to the stable or metastable states of the materials system. The energy difference between initial state and the saddle point with the lowest possible energy barrier on a PES, which corresponds to the highest energy level along the minimum energy path (MEP), gives the estimate of the transition rate constant. The lower the energy difference is, the easier or faster the transition could be. Most of these pathway and saddle-pont search methods, which will be summarized in Section 2.2 and Section 2.3, rely on an initial guess of the transition path from the initial state (or phase) to the final one. The search usually is a local refinement process of which the final path passes through the saddle point with the lowest possible energy barrier. Thus the accuracy of these methods sensitively depends on the initial guesses of the paths. Existing methods give the initial guesses by either simple linear interpolation of atoms' positions or case-by-case empirical approaches. New approaches which systematically provide initial guesses that are reasonably close to the MEP are needed. In the remainder of this chapter, a summary of existing transition path and saddle point search methods and molecular scale geometric modeling techniques is presented.

2.1 Geometric Modeling of Molecular Structures

As part of research efforts in computer aided molecular design, modeling of geometry and topology of molecular structures has attracted researchers' attention. More detailed introduction and analysis of the Periodic Surface (PS) model, which provides a convenient method for representing porous, repetitive structures, is presented in [7] as well as Chapter 3. A feature based approach to construct crystal structures based on loci surfaces using the PS model was proposed [8]. Additionally, based on PS models, reconstruction of loci surfaces from crystals [9], complexity control [10], and Minkowski sum [11] were studied.

In addition to the PS model, other methods have also been investigated and shown to be effective. Edelsbrunner developed a novel method for modeling smooth surfaces based on skins specified by a set of weighted points [12]. Similarly, a method for reconstructing surfaces from a finite set of points was also proposed [13]. Bajaj et al. represented the surface boundary of molecules using a set of non-uniform rational B-spline patches [14]. Other efforts in geometry modeling include the construction of quality meshes for implicit salvation models of biomolecular structures [15] and computation and triangulation of the molecular surface of a protein with beta shapes [16,17,18]. Topology of ribbons [19], frequently used for modeling of DNA and proteins, was described in terms of the "knottiness" or link between two curves. An approach for computing the Euclidean Voronoi diagram for spheres [20] has also been presented. The

Voronoi diagram was further used as a tool for meshing of particle systems within bounded regions [21].

2.2 Transition Path Search

In order to determine the magnitude of the activation energy barrier during transition, we must first find the transition path. Transition path search methods are classified either as chain-of-states methods, including nudged elastic band (NEB) and string methods, or as one of the other methods. Chain-of-states methods rely on a collection of images that represent intermediate states of the atomic structure as it transforms from initial to final configurations along the transition path. These discrete states are chained to each other using interpolation after the search converges, and the transition pathway and saddle point are obtained. Chain-of-states methods work best on surfaces that may have more than one saddle point; when there is more than one transition pathway, these methods converge to the path which is closest to the initial guess. The most common and well developed of these methods is the NEB [22], which relies on a series of images, or points in the configuration space corresponding to intermediate states, connected by springs. To increase resolution at the region of interest (ROI) and accuracy of saddle point energy estimates, the NEB method omits the perpendicular component of the spring force, as well as the parallel component of the true force due to the gradient of potential energy. The purpose of the springs is to keep the images evenly distributed on the transition path. In some cases, this method produces paths with unwanted kinks when the potential energy changes quickly, or it may not have any images that are directly on the saddle point. The improved tangent NEB [23] and doubly nudged elastic band [24] methods reduce the appearance of kinks by generating a better estimate of the tangent direction of the path at each image and re-introducing a perpendicular spring force component, respectively. The estimate of tangent direction is improved by using only one neighbor rather than two neighbors and a central finite difference approximation as with the original NEB. Free-end NEB [25] only requires knowledge of either the initial or final state, rather than both, and climbing image NEB [26] allows the image with the highest energy to climb in order to locate the saddle point. Eigenvector following optimization can be applied to the result of NEB to locate actual saddle points, and results can be improved further by introducing energy-weighted or gradient-weighted adaptive spring constants that increase the resolution of the ROI [27].

String methods [28,29] represent the transition path continuously as Splines that evolve and converge to the MEP in two steps when subjected to perpendicular forces. The curve is discretized as a set of points and solved by standard ODE solvers in the evolution step; these points are then redistributed along the string based on parameterization constraints and Spline interpolations in the re-parameterization step. As opposed to NEB, the number of points used in the String method can be modified dynamically. The Growing String method [30] takes advantage of this by starting with points at the reactant and product, and then adding points which meet at the saddle point. The quadratic String method [31] is a variation that uses a multi-objective optimization approach. A local quadratic approximation of the PES is made and searched using the quasi-Newton technique.

Methods that are not classified as chain-of-states include the path sampling and averaging accelerated Langevin dynamics method [32] and the conjugate peak refinement method [33], which finds saddle points and the MEP by searching the maximum of one direction and the minima of all other conjugate directions iteratively. The Hamilton-Jacobi method [34] relies on the solution of a Hamilton-Jacobi type equation to generate the MEP.

2.3 Saddle Point Search

Instead of searching the complete MEP, saddle point search methods only locate the saddle point on the MEP. They are categorized into local and global search methods. One of the original local methods is the automated surface walking algorithm [35,36]. It is based on eigenvectors of the Hessian matrix with local quadratic approximations of the PES. The Hessian matrix is updated iteratively similar to the quasi-Newton method, and one of the active coordinates is scaled so that the Hessian eigenvalues lie in a required range. The more recent ridge method [37] uses a pair of images to search for the saddle point, with the direction of the connecting line between the two images constrained such that the pair is kept on the ridge. Also using two images, the dimer method [38] has a fixed distance between them. The pair first moves uphill in the translation step, and then rotates towards the lowest curvature mode of the PES in the rotation step. Reduced Gradient Following [39] and Reduced Potential Energy Surface Model [40] methods use intersections of zero-gradient curves and surfaces, with saddle point search occurring within the subspace of these curves or surfaces. Finally, the Synchronous Transit method [41] estimates the transition state by minimizing the interpolated inter-atomic distances and refines the saddle point estimate through conjugate gradient optimization.

Local search methods may locate the saddle point which does not have the maximum energy on the MEP if there are multiple saddle points. Global search methods have the advantage that the saddle point with the maximum energy is located if the search converges. The Dewar-Healy-Stewart method [42] searches for the saddle point by iteratively reducing the distance between reactant and product images. The Activation-Relaxation technique [43] can travel between many saddle points using a two step process; an image first jumps from a local minimum to a saddle point, and then back down to another minimum. This approach allows for movement between many saddle points without the knowledge of the final product. The Step and Slide method [44] uses an image from the initial and final state. Energy levels of each are increased gradually, and the distance between them is minimized while remaining on the same isoenergy surface. The interval Newton's method [45] is capable of finding all stationary points by solving the equation of vanishing gradient.

The proposed geometry guided approach in this thesis is to provide an initial guess of the transition pathway that is reasonably close to the MEP in order to accelerate the searching of the chain-of-state methods, particularly the widely used climbing image NEB method. The geometry of crystals is constructed by a periodic surface model. The initial guess is computed by the metamorphosis of the surface model.

2.4 Particle Swarm Optimization and BFGS Algorithm

Particle Swarm Optimization (PSO) was first proposed by James Kennedy and Russell Eberhart in 1995 [46]. It is based on the behavior of groups of birds or insects, and the movement of the entire swarm towards an "optimum" location, such as a food source, even when some individual members may not know the exact location. The movement updates of each individual in the group are based on its own velocity and position, as well as its on best previous location and the best previous location which was found by the swarm as a whole. With the classical PSO algorithm, this approach results in a search of a relatively large region and convergence of all the individuals to one area which was found to be optimal. Compared to other optimization techniques, it is relatively simple and resistant to local optima. By adding more individuals to the swarm and increasing the size of the initial search space, the search can be very comprehensive and convergence to a global optimum is still achieved. By varying some of the parameters discussed in chapter 4, individual particles can also be forced to favor either their own previous best position or the swarm's previous best, allowing for customization based on the application.

Mathematically, the velocity update for N particles with I iterations is expressed as

$$V_j^{(i)} = \vartheta V_j^{(i-1)} + c_1 r_1 [P_{best,j} - X_j^{(i-1)}] + c_2 r_2 [O_{best} - X_j^{(i-1)}]$$

where $j = 1,2,...,N$, $i = 1,2,...,I$, and $V_j^{(i)}$ is the velocity of particle j at iteration i. ϑ, c_1, and c_2 are constants, r_1 and r_2 are random numbers between 0 and 1, $P_{best,j}$ is the best previous position of particle j, O_{best} is the neighbors' previous best position, and X_j is the location of particle j. Each iteration also updates the positions as

$$X_j^{(i)} = X_j^{(i-1)} + V_j^{(i)}$$

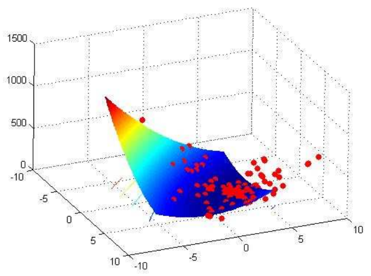
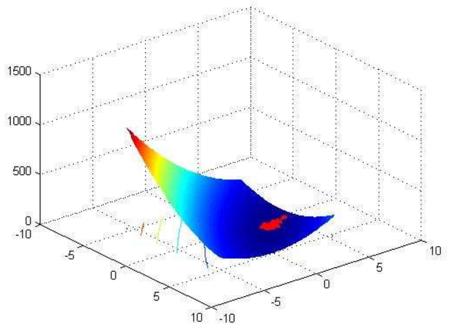
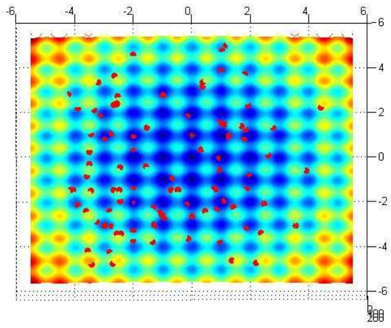
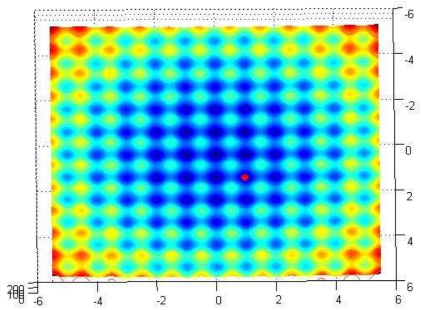
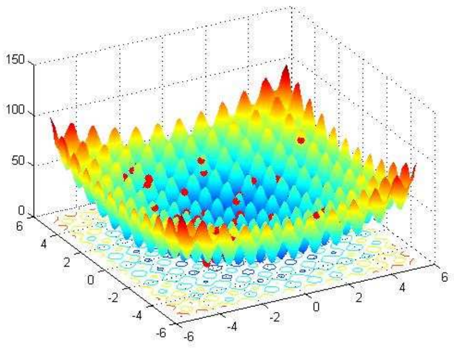
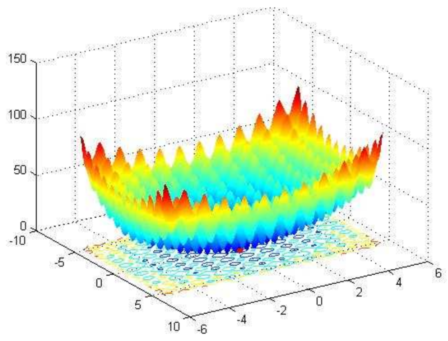
With this original method, usually every particle in the search space is considered a neighbor, so O_{best} is the best position of any particle in the population. When searching for a global minimum, the best position is defined as one where the value of the objective function is the lowest. A pseudo-algorithm for the PSO searching process is shown in Table 1. This pseudo-code demonstrates the procedure using a generic objective function and considering all particles as neighbors. The objective function can be of any degree, although higher degree functions generally require a longer computation time to find the global optimum.

Table 1: Pseudo-code for the standard PSO algorithm

```
**INPUT**: number of particles $\{N\}$, upper and lower search space boundaries $\{b_u, b_l\}$, objective function $\{f(x_1,\ldots,x_N)\}$, maximum number of iterations $\{Maxi\}$, constants for local and global directions $\{c_1, c_2\}$, movement restriction $\{J\}$, convergence threshold $\{epsilon\}$
**OUTPUT**: location of global optimum $\{x_1,\ldots,x_N\}$
FOR $i = 1:N$
    $X_i = rand()*(b_u - b_l) + b_l$
    $pbest_i = X_i$
    $V_i = 0$
    $f_i = f(X_i)$
END
$obest = \arg\min_i(f)$
$fbest = \min_i(f)$
FOR $i = 1:Maxi$
    FOR $j = 1:N$
        $newV_j = \vartheta * V_j + c_1 * rand()*(pbest_j - x_j) + c_2 * rand()*(obest - x_j)$
        $newX_j = X_j + newV_j$
        $newf_j = f(X_j)$
    END
    $X = newX$
    $V = newV$
    $obest = \arg\min_{j,obest}(newf_j, fbest))$
    FOR $n = 1:N$
        IF $newf_n < f_n$
            $pbest_n = X_n$
        END
    END
    $f = newf$
    IF $\min(f_i) < epsilon$
        BREAK
    END
END
```

With a sufficiently large N, this PSO technique is a powerful tool for finding the minimum with a variety of functions. Because of the random initial distribution of particles, much of the search space is explored, and velocity updates based on neighboring particles increase the likelihood of convergence. However, the algorithm is not robust for functions with many similar local minima, and oftentimes fails to converge to the global optimum location. Table 2 shows a visual demonstration of the initial random distribution of particles as well as the distribution after convergence for two different functions. In this case a population of 100 particles was used. It can be seen that the algorithm works well for the relatively simple Booth function, as all the particles quickly converged to the global minimum. However, in the case of the more complex Rastrigin function, the particles converged to one of the local minima rather than the global minimum in the center. As will be shown in section 4.4, the success rate for this function is particularly low. This tendency to converge to a local minimum on complex functions is one of the main drawbacks of the standard PSO algorithm. Additionally, it does not provide any information about the landscape besides the location that was obtained, so it can be difficult to discern whether the optimum that was found is local or global. To address these issues, we propose the following two novel optimization techniques based on the original PSO method.

Table 2: Basic PSO

Function	Initial (Random) Distribution	Final Distribution
Booth		
Rastrigin		
Rastrigin (Iso)		

Improvements and additions have been made by various research groups since the original publication. For the most part, improvements focus on the particles' tendency to stagnate in some areas of the search space, and to clump together prematurely when a potential global optimum is found. The Collision-Avoiding Swarms algorithm [47] presents an application for improvised music in which particles are attracted to the center of mass where the global optimum is most likely located, but at the same time experience a repulsive force to prevent collisions with other particles. Similarly, Self-Organized Criticality [48] assigns some or all particles a criticality value, which increases when it gets within a threshold distance from another particle. If a particle's criticality becomes too high, the criticality value is distributed between its' nearest neighbors. Dissipative PSO [49] introduces additional "chaos" in the optimization algorithm to increase the velocity of particles for some iterations and prevent stagnation. The concept of sub-swarms with different variations [50,51] is introduced to split the initial swarm into smaller groups that exchange information among each other and rearrange themselves frequently. Hierarchical grouping of particles according to quality was also investigated [52]. Incorporating some ideas from ant colony optimization, the Estimation of Distribution PSO [53] uses all the particles' personal best locations to guide the swarm towards the most promising regions of the search space. Finally, the PSO with Spatial Particle Extension [54] is another proposed method to prevent particles from clustering together by bouncing them off each other if the distance between particles is too small. The PSO method and its extensions have been tested on a number of different test functions up to relatively high dimensions [55].

In order to integrate the PSO algorithms discussed in Chapter 4 with the Quantum Espresso software package, we must first investigate the native methods employed within the software. Quantum Espresso uses the quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimization method for structure relaxation. When a previously known arrangement of atom positions in a crystal structure is provided, it must first be "relaxed" into a minimum energy position before transition pathway search can begin. Each atom moves to a location that represents an overall minimum energy position on the PES. In this case the PES is the objective function and each atom location is a parameter. Therefore, the dimension of the objective function depends on the number of atoms in the structure, and the overall energy of the structure depends on the atom locations. Thus optimization must be performed to move each atom into a region that minimizes energy for the system overall.

The BFGS method iteratively updates each atom's position using a quasi-Newton approximate Hessian method. The Hessian is initially approximated as an identity matrix, and is subsequently updated in each iteration. The inverse Hessian is required to update positions, and is evaluated according to

$$B_{i+1}^{\ -1} = \frac{B_i^{-1} + (\mathbf{S}_i^{\ T}\mathbf{y}_i + \mathbf{y}_i^{\ T} B_i^{-1}\mathbf{y}_i)(\mathbf{S}_i\mathbf{S}_i^{\ T})}{(\mathbf{S}_i^{\ T}\mathbf{y}_i)^2} - \frac{B_i^{-1}\mathbf{y}_i\mathbf{S}_i^{\ T} + \mathbf{S}_i\mathbf{y}_i^{\ T} B_i^{-1}}{\mathbf{S}_i^{\ T}\mathbf{y}_i}$$

where

$$\mathbf{S}_i = \alpha_i \mathbf{p}_i$$

$$\alpha_i = \mathbf{x}_{i+1} - \mathbf{x}_i$$

$$\mathbf{y}_i = \mathbf{g}_{i+1} - \mathbf{g}_i$$

and

$$\mathbf{p}_i = -B_i^{-1}\mathbf{g}_i$$

B represents the approximate Hessian matrix, $\vec{g}_i$ is the gradient of the objective function, and $\vec{x}_i$ is the position of atom i. Position updates continue according to

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_i$$

until the convergence threshold of the objective function value is reached.

In the following chapters, improvements to the NEB method are proposed using geometric modeling via periodic surfaces. Additionally, in chapter 4, two extensions to the basic PSO algorithm are presented that improve accuracy and convergence rate, and a method of incorporating these PSO algorithms with phase transition simulation is outlined.

3. CORRESPONDENCE AND TRANSITION PATH SEARCH

In this chapter the methods for periodic plane correspondence and transition path search, including some situational techniques that can be applied, are explained. First an overview of the periodic surface model is given, followed by discussion of crystal construction with foci surfaces in section 3.1. Then atom correspondence search is presented in section 3.2, along with the concepts of atom correlation and classification of positions. For the remainder of the chapter, section 3.3 covers transition path search with plane correspondence by minimum potential, including the simulated annealing pair-searching algorithm and plane constraints for strongly bonded pairs.

The periodic surface model has the implicit form and is defined as

$$y(\mathbf{r}) = \sum_{l=1}^{L}\sum_{m=1}^{M} m_{lm} \cos\left(2pk_l(\mathbf{p}_m^T \cdot \mathbf{r})\right) = 0 \qquad (1)$$

where k_l is the scale parameter, $\mathbf{p}_m = \left[a_m, b_m, c_m, a_m\right]^{\mathrm{T}}$ is a basis vector, such as one of

$$\{\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_5, \mathbf{e}_6, \mathbf{e}_7, \mathbf{e}_8, \mathbf{e}_9, \mathbf{e}_{10}, \mathbf{e}_{11}, \mathbf{e}_{12}, \mathbf{e}_{13}, \mathrm{K}\} =$$
$$\left\{ \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \begin{bmatrix}1\\0\\0\\1\end{bmatrix} \begin{bmatrix}0\\1\\0\\1\end{bmatrix} \begin{bmatrix}0\\0\\1\\1\end{bmatrix} \begin{bmatrix}1\\1\\0\\1\end{bmatrix} \begin{bmatrix}1\\0\\1\\1\end{bmatrix} \begin{bmatrix}0\\1\\1\\1\end{bmatrix} \begin{bmatrix}1\\1\\1\\1\end{bmatrix} \begin{bmatrix}1\\-1\\0\\1\end{bmatrix} \begin{bmatrix}-1\\0\\1\\1\end{bmatrix} \begin{bmatrix}0\\1\\-1\\1\end{bmatrix} \begin{bmatrix}1\\-1\\1\\1\end{bmatrix} \begin{bmatrix}-1\\1\\1\\1\end{bmatrix} \begin{bmatrix}1\\1\\-1\\1\end{bmatrix} \mathrm{K} \right\}$$

which represents a basis plane in the Euclidean space R^3, $\mathbf{r} = \left[x, y, z, w\right]^{\mathrm{T}}$ is the location vector with homogeneous coordinates, and m_{lm} is the periodic moment. We assume $w = 1$ if not explicitly specified. The degree of $y\left(\mathbf{r}\right)$ in Eqn.(1) is defined as the number of unique vectors in the basis vector set $\{\mathbf{p}_m\}$. The scale of $y\left(\mathbf{r}\right)$ is defined as the number of unique scale parameters in $\{k_l\}$. We can assume scale parameters are natural numbers ($k \in$ ¥).

Fig. 2 lists some examples of periodic surface models. Triply periodic minimal surfaces, such as P-, D-, G-, and I-WP cubic morphologies that are frequently referred to in chemistry and polymer literature, can be adequately approximated. Besides the cubic phase, other mesophase structures such as spherical micelles, lamellar, rod-like hexagonal phases can also be modeled. The lamellar structure, for example, can be represented as a periodic surface model using the equation

$$\cos(2\pi z) = 0 \tag{2}$$

and the P-structure is described using

$$\cos(2\pi x) + \cos(2\pi y) + \cos(2\pi z) = 0 \tag{3}$$

Equations corresponding to the other structures in Fig. 2 are more involved. They are discussed in greater detail in [7].

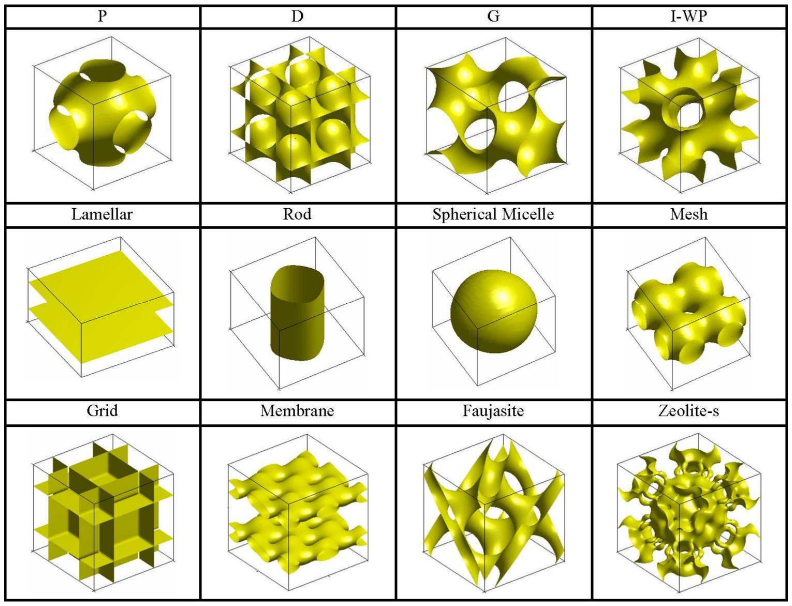


Fig. 2: Periodic surface models of cubic phase and mesophase structures

The searching process in computer-aided transition pathway design starts with a unit cell of a crystal material in its initial state before it undergoes phase transition. The desirable locations of the atoms that make up this unit cell are known, and a loci surface model is reconstructed. Similarly, loci surface model reconstruction is also used for the final state to which the material will transition. The next step is to find intermediate steps between the known initial and final states. Using the atoms in the unit cell, the location of each atom in the initial state is compared to all the atoms in the final state. The correspondence between the states is determined based on the minimum distance approach or the minimum energy change, which will be discussed in Section 3.2 and Section 3.3, respectively. Once it is known to which location each atom transitions, interpolation of corresponding PS models is used to find the atom locations at intermediate states. At each intermediate state, interpolated loci surfaces are used to model the geometry. Particularly for crystals, the simplest loci surfaces are periodic planes. This information about the geometric transition process of the unit cell, as a more accurate initial guess of the transition path, can be fed into the transition pathway search methods.

3.1 *Crystal Construction by Loci Surfaces*

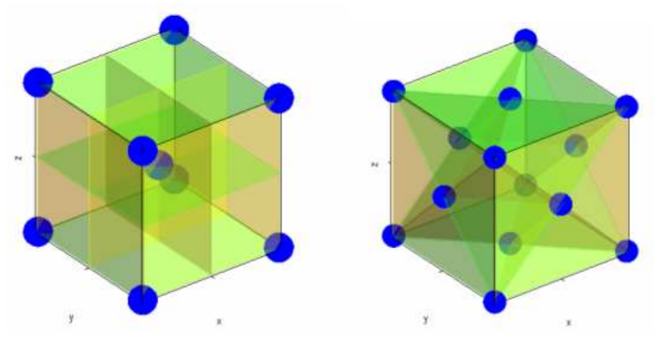
A process of tiling by intersection as described in [8] can be used to construct crystal structures. They are built with 14 Bravais lattices, each of which can be constructed via intersections of periodic surfaces. For three periodic surfaces $y_1(\mathbf{r}) = 0, y_2(\mathbf{r}) = 0, y_3(\mathbf{r}) = 0$, the intersection is found by solving $y(\mathbf{r}) = y_1^2(\mathbf{r}) + y_2^2(\mathbf{r}) + y_3^2(\mathbf{r}) = 0$. This provides a method for generating each of the points in a lattice. For instance, Fig. 3 shows a body centered and a face centered cubic structure. They are generated by

$$\begin{aligned}&\cos^2\left(\pi\left(x - y + 0.5\right)\right) + \cos^2\left(\pi(x + y + 0.5)\right) + \\ &\cos^2\left(\pi(y + z + 0.5)\right) = 0\end{aligned} \tag{4}$$

and

$$\begin{aligned}&\cos^2\left(\pi\left(x + y + z\right)\right) + \cos^2(\pi(-x + y + z)) + \\ &\cos^2(\pi(-x - y + z)) = 0\end{aligned} \tag{5}$$

respectively. The markers in the figure indicate atom positions generated by intersections of periodic surfaces. In the same way, all types of lattices can be constructed.



(a) body-centered cubic (b) face-centered cubic

Fig. 3: Body centered and face centered cubic structures constructed by loci periodic surfaces

The most generic approach to reconstruct loci surface models from crystals is by constructing y-z, x-z, and x-y planes for each atom. Given a unit cell with the size of a, b, and c in the respective x, y, and z direction, the y-z, x-z, and x-y planes that go through the origin (0,0,0) have the respective basis vectors

$$\begin{cases} \mathbf{p}_{yz}(0) = [1,0,0,a/2] \\ \mathbf{p}_{xz}(0) = [0,1,0,b/2] \\ \mathbf{p}_{xy}(0) = [0,0,1,c/2] \end{cases} \tag{6}$$

and the respective scale parameters

$$\begin{cases} k_x = 1/(2a) \\ k_y = 1/(2b) \\ k_z = 1/(2c) \end{cases} \tag{7}$$

If an atom in the unit cell has the coordinates x, y, and z, then the respective basis vectors for the y-z, x-z, and x-y planes that go through the atom are

$$\begin{cases} \mathbf{p}_{yz}(x) = [1,0,0,a/2+x] \\ \mathbf{p}_{xz}(y) = [0,1,0,b/2+y] \\ \mathbf{p}_{xy}(z) = [0,0,1,c/2+z] \end{cases} \tag{8}$$

with the same scale parameters as in Eqn.(7).

Obviously, when special knowledge about atoms is available, the periodic planes to construct atoms are not necessarily y-z, x-z, or x-y planes, such as the ones in Fig. 3(b). The number of planes can be reduced because of the correlation between atoms. Similar to Eqn.(6), the information required to build a plane is the normal direction of the plane and its distance between the atom and the new origin of reference along the normal direction. For an atom in the unit cell with the coordinates x, y, and z, the basis vector of a periodic plane with the normal vector $\hat{n} = (n_x, n_y, n_z)$ (where $n_x^2 + n_y^2 + n_z^2 = 1$) is either $\mathbf{p}(x,y,z) = [n_x, n_y, n_z, a/2 - (n_x x + n_y y + n_z z)]$, $\mathbf{p}(x,y,z) = [n_x, n_y, n_z, b/2 - (n_x x + n_y y + n_z z)]$, or $\mathbf{p}(x,y,z) = [n_x, n_y, n_z, c/2 - (n_x x + n_y y + n_z z)]$ corresponding to the rotated plane with respect to *x*-, *y*-, or *z*-axis. The respective scale parameter for the plane is $k = n_x / (2a)$, $k = n_y / (2b)$, or $k = n_z / (2c)$.

One important question that needs to be answered for the transition searching process is how atoms in initial and final states are corresponding to each other. Searching the correspondence between atoms is described in Section 3.2.

3.2 Searching Correspondence of Atoms

In general, there are two steps in modeling surface morphing. First, the location correspondence of atoms is identified. Then the PS models of the corresponding atoms are paired and the interpolation is made between them. To compare atom locations between the initial and final states, we may use a matrix form for the locations, with rows 1, 2, and 3 containing the x, y, and z coordinates, respectively. For example, a cubic structure with one corner at (0,0,0) and size of a, b, and c is represented by the matrix

$$\begin{bmatrix} 0 & a & a & 0 & 0 & a & a & 0 \\ 0 & 0 & b & b & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & c & c & c & c \end{bmatrix}$$

It is assumed that each atom will transition to the nearest position in the final state. Data for atom locations in the final state is listed in a matrix with identical dimensions. Starting with the first column in the matrix of the initial phase, the Euclidean distance is calculated between this location and each location in the matrix of the final phase. The process is then repeated for all other columns in the initial matrix. The correspondence between locations in the initial and final phases is determined based on the minimum distance between them. That is, if $[q_1,\dots,q_n]$ is the initial matrix containing n locations and the final matrix is $[q_1^{'},\dots,q_n^{'}]$, then the distance d_{ij} between the ith location in the initial matrix and the jth in the final one is $d_{ij} = |q_i - q_j^{'}|$ where $1 \le i \le n$ and $1 \le j \le n$. For the ith location at the initial stage, the corresponding i_Fth location at the final stage is determined by

$$i_F = \arg\min_{1 \le j \le n} d_{ij} \tag{9}$$

However, for complex crystal structures it becomes inconvenient to track all points individually. Inaccuracy can be introduced if there is a significant amount of rotation or scaling in the crystal, as the assumption may no longer be valid that each atom moves to the nearest position. Therefore, some improvements and simplifications can be introduced for certain structures. These include the classification approach described in

Section 3.2.1 and the correlation approach introduced in Section 3.2.2.

3.2.1 *Classification of Positions*

In many crystal structures, more than one type of element is present in the unit cell. In these cases it is not necessary to compare the locations of all atoms in the initial and final states because an atom of one element cannot move into a location occupied by a different element. The data in the location matrices needs to be sorted so that the atoms of each element are grouped together. In general, if certain atoms are not likely to be in certain positions, those positions can be excluded in the pair-wise comparison. The available positions are classified and grouped into several subsets. For example, in a body centered cubic structure, the first eight columns of the matrix can represent the corner atoms which are all the same element, and a final ninth column would represent the location of the central atom. Each atom in the initial configuration would then only be compared to atoms of the same element in the final phase, reducing the amount of computation needed in multi-element structures.

Suppose that there are a total of T different types of elements. The column indices of the location matrix can be grouped into T subsets as

$$(1,\ldots,n_1)(n_1+1,\ldots,n_2)\cdots(n_{t-1}+1,\ldots,n_t)\cdots(n_{T-1}+1,\ldots,n)$$

The computation of minimal distance for type t element then is based on

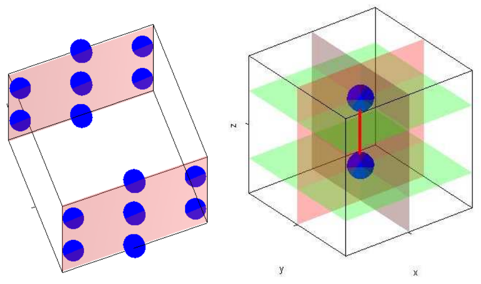
$$i_F = \arg\min_{n_{t-1}+1\le j\le n_t} d_{ij} \tag{10}$$

instead of Eqn.(9).

3.2.2 *Correlation of Atoms*

Depending on the types of bonds in the crystal, there may be groups of atoms that remain equidistant from each other on the same plane throughout the phase transition process. With graphite, for example, the bonds between carbon atoms along each plane are stronger than the bonds that connect planes to each other. The weaker bonds are more likely to separate, leaving the planes intact. This type of property can be taken into consideration when modeling the phase transition process. Atoms that are located on the same plane and remain in the same position relative to each other do not have to be considered individually during the process outlined in Section 3.2.1. Only the position of one of the atoms on the plane must be found, and the rest are placed in the same positions with respect to the coordinates of the first. This reduces the amount of computation when comparing atom coordinates because the corresponding position must be found for only one reference atom on the plane.

In the graphite example, the structure can be modeled with hexagonal unit cells where atoms in the individual planes are connected with covalent bonds, while the planes are connected to each other by the van der Waals force. This indicates that atoms which are in the same plane are likely to remain on that plane. We call this special case *face correlation*. An example of a hexagonal structure is shown in Fig. 4(a) where the colored surfaces represent planes along which the atoms are covalently bonded. We can take advantage of these characteristics when modeling the structure by reducing the number of periodic surfaces required to construct it. Normally each of the 14 atoms in the unit cell of Fig. 4(a) would be represented by three perpendicular planes, meaning that interpolation must be performed on 42 individual periodic surfaces. However, because the atoms in each layer have a common periodic surface, this number is reduced to 30. Two unique surfaces are required for each point, with the third being the common planes on the top and bottom of the hexagonal unit cell, shown in Fig. 4(a).



(a) (b)
Fig. 4: Correlation of atoms (a)Hexagonal unit cell with face correlation, (b)Two strongly bonded atoms with edge correlation

In cases where only two atoms have a bond that is stronger than other bonds in the structure, these atoms are less likely to break apart during phase transition. The number of surfaces required for modeling can be reduced again because of shared intersection planes. The two planes that intersect to form the line along which the atoms are bonded are used to generate the locations of both atoms. Thus, instead of using six planes to model these two atoms, the number can be reduced to four. If the two atoms in Fig. 4(b) have a strong bond and can be assumed to remain in the same position relative to each other, they can be modeled using the four planes shown. The two vertical planes are common to both atoms. Their intersection represents the line along which the atoms are bonded. In addition, the two horizontal planes are used to define each atom's position along the z-axis. This special case of periodic surfaces is called *edge correlation*.

In summary, we find correspondence of atoms between initial and final states so that the respective periodic planes can be constructed. The interpolation of the planes then locates intermediate positions of atoms during the transition process. If all three planes for each atom are fully constructed, the correlation and energy exchange between atoms are ignored. Pair-wise comparison between individual atoms is used in searching correspondence, which is purely based on geometry. Instead, if the number of planes is reduced because of face correlation or edge correlation, the interaction among atoms is implicitly considered. Geometry and physics become more integrated. Yet, this atom correspondence approach assumes corresponding planes between initial and final states only translate during the transition process, whereas the rotation of planes is not considered. For instance, in the most general case, all planes are in either y-z, x-z, or x-y direction. A y-z plane in the initial state only translates to a y-z plane in the final state, similar for the other two. More importantly, when the number of planes increases, the exhaustive searching method of atom correspondence becomes combinatorially expensive. In Section 3.3, we present a different approach to find the correspondence of periodic planes directly by a heuristic searching method without the need of computing the correspondence of atoms.

3.3 Correspondence of Periodic Planes by Minimum Potential Energy Change

Different from the atom correspondence method in Section 3.2 where each plane will simply move to the nearest available position, an alternative approach is to find the correspondence of planes directly by considering the total potential energy change of the system. We define the total potential energy change as a function of both displacement and rotation of each plane. The pair-wise correspondence between the initial and final planes is found by searching the minimum potential change. This method yields better results in more general structures where the planes undergo both rotation and translation. Since searching the global minimum of potential energy change has combinatorial complexity, heuristic optimization methods can be used for large systems. Here, we use a Monte Carlo simulation or simulated annealing (SA)-like algorithm. The potential energy change is defined in Section 3.3.1, and the SA search algorithm is described in Section 3.3.2.

3.3.1 Translational and Rotational Potential Change

As illustrated in Fig. 5, plane i used in the construction of a unit cell in the initial state is represented by a point and a vector, $\vec{a}_i$ and $\vec{p}_i$ respectively. If there are two or more atoms located on the plane, $\vec{a}_i$ can be placed at the center of the convex polygon formed by the atoms; otherwise it is simply placed at the location of the atom. $\vec{p}_i$ is a unit vector indicating the normal direction of the plane. Similarly, points $\vec{b}_i$ and vectors $\vec{q}_i$ are placed on the unit cell for the final state.

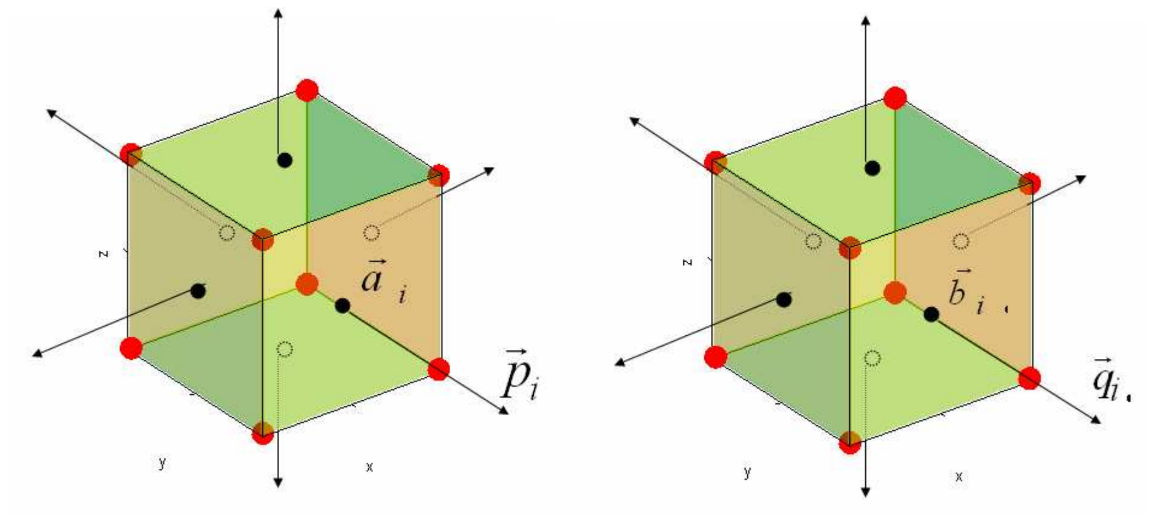


Fig. 5: Correspondence between locations and directions of planes

The points $\vec{a}_i$ and $\vec{b}_i$ are used to calculate the displacement of the planes. We use the Euclidean distance formula to find the distance d_i between $\vec{a}_i$ and $\vec{b}_i$, which is then used to find the translational potential change $\Delta V_i(d_i)$ between the two locations for pair i, given by

$$\Delta V_i(d_i) = s[\frac{d_i^2}{(d_{\max})}] \tag{11}$$

where s is a constant coefficient and $d_{\max}$ is the largest possible distance between any two planes.

In addition to the translational potential change, a rotational potential change is also defined for the transition of each plane. The angle θ_i between $\vec{p}_i$ and $\vec{q}_i$ is found and used to calculate the rotational potential change $\Delta U_i(q_i)$, given by

$$\Delta U_i(q_i) = c[\frac{1 - \cos(q_i)}{2}] \tag{12}$$

where c is a constant coefficient. Through experimentation with different values, $c = s = 300$ was found to be the most effective setting. The ratio of c to s is the most important factor as it determines whether rotation or translation is preferred in the optimization. By settting the two values equal to each other, the correspondence search treats both types of movement equally. Because the translational and rotational potential change terms are normalized, the numerical value of c and s is not important for an exhaustive search. However, when using an optimization algorithm as discussed in the next section, higher values allow for faster convergence when compared to lower values. By running trials with a range of different numbers, it was found that $c = s = 300$ yields the best combination of convergence speed and accuracy. $\Delta U_i(q_i)$ is added to $\Delta V_i(d_i)$ to receive the combined potential change for the plane's transition. In a structure with n planes, the total potential change is obtained by the summation of all planes as

$$\Delta W_{total} = \sum_{i=1}^{n}(\Delta U_i + \Delta V_i) \tag{13}$$

Searching the correspondence of periodic planes is to find an arrangement with the minimum total potential change between the initial and final states. In each iteration, $\vec{a}_i$ and $\vec{p}_i$ remain unchanged, but they are used in combination with a different pair of $\vec{b}_i$ and $\vec{q}_i$ to find the total potential change. The combination that yields the lowest ΔW_{total} is used to determine to which location each plane transitions. After the correspondence between planes is determined, each of the atom locations in the final state is found using the intersection of the planes.

When the total number of planes is low such as in simple crystals, all combinations can be checked. For complex crystal structures with a large number of basis atoms in one unit cell, the effects of combinatorial complexity make it highly impractical to check all possible combinations. Even a structure with ten plane locations presents billions of possibilities. It becomes computationally expensive to go through more than a few thousand iterations, so a heuristic global optimization approach is preferred. The algorithm discussed in the following section provides a method of optimizing the solution without searching through all possible combinations. Although individual iterations of the algorithm are more involved than the simple exhaustive searching technique, the overall searching process is less computationally expensive for cases where the structure is more complex.

3.3.2 Simulated Annealing (SA) Algorithm

In order to generalize the method and make it applicable to a wider range of structures, we use a SA-like optimization method to find the match with the minimum total potential change. The pseudo-code of the SA algorithm is listed in Table 3. In each iteration, pairs of planes in the final state are switched. Two randomly chosen $\vec{q}_i$ and $\vec{b}_i$ values are switched and a new total potential change $\Delta W_{total,new}$ is found using the new arrangement. Using the Metropolis criterion, if $(\Delta W_{total,new} - \Delta W_{total}) < 0$, the switch is accepted. Otherwise, the switch may still be accepted, but with a certain probability. That is, a random number $u \in [0,1)$ is generated. If $u \le \exp\left((\Delta W_{total} - \Delta W_{total,new}) / T\right)$ where T is a temperature variable, then the switch is accepted. Otherwise, it is rejected. The value of T is decreased over time, after either every iteration or every few iterations, to simulate cooling of the material. Table 3 lists the pseudo-code of the algorithm.

3.3.3 Plane Constraints for Strongly Bonded Pairs

In the cases where a pair of atoms has a bond that is much stronger than other bonds in the structure, as discussed in section 3.2.2, some constraints must be placed on the corresponding planes to prevent the bond from breaking or elongating. Using Fig. 5 as reference, the two horizontal planes must remain equidistant from each other throughout the transition. The distance between the two planes is kept unchanged and specified as

$$(\vec{a}_1 - \vec{a}_2) \cdot \vec{p}_1 / |\vec{p}_1| = (\vec{b}_1 - \vec{b}_2) \cdot \vec{q}_1 / |\vec{q}_1| \tag{14}$$

where $\vec{a}_1$ and $\vec{p}_1$ correspond to one plane and $\vec{a}_2$ corresponds to the other at the initial state, whereas $\vec{b}_1$, $\vec{q}_1$ and $\vec{b}_2$ correspond to the final state.

Table 3: Pseudo-code of the algorithm to search correspondence of planes with minimum potential change

```
INPUT: initial points $\{\vec{a}_i\}$, final points $\{\vec{b}_i\}$, initial vectors $\{\vec{p}_i\}$, final vectors $\{\vec{q}_i\}$, temperature $T$
OUTPUT: Combination of switches that yields the lowest $\Delta W_{total}$
$size$ = number of planes in the structure;
$\Delta T$ = Interval of temperature change;
$\Delta W_{total} = \sum_{i=1}^{size} (\Delta U_i + \Delta V_i)$;
WHILE ($T > 0$)
    $m$ = random integer between 1 and $size$
    $n$ = random integer between 1 and $size$
    $q_1^{(old)} = q_m$; $q_2^{(old)} = q_n$; $b_1^{(old)} = b_m$; $b_2^{(old)} = b_n$;
    $q_n = q_1^{(old)}$; $q_m = q_2^{(old)}$; $b_n = b_1^{(old)}$; $b_m = b_2^{(old)}$;
    $\Delta W_{total,new} = \sum_{i=1}^{size} (\Delta U_i + \Delta V_i)$;
    IF $(\Delta W_{total,new} - \Delta W_{total}) > 0$
        $u$ = random number between 0 and 1;
        $g = \exp\left(-(\Delta W_{total,new} - \Delta W_{total}) / T\right)$;
        IF $u > g$
        $q_m = q_1^{(old)}$; $q_n = q_2^{(old)}$; $b_m = b_1^{(old)}$; $b_n = b_2^{(old)}$;
            $\Delta W_{total,new} = \Delta W_{total}$;
        END
    END
    $\Delta W_{total} = \Delta W_{total,new}$;
    $T = T - \Delta T$;
END
```

Additionally, the angle between the two horizontal planes may remain constant. That is, for each pair of planes i and j, the condition

$$\vec{p}_i \cdot \vec{p}_j = \vec{q}_i \cdot \vec{q}_j \tag{15}$$

must be satisfied. For the searching process outlined in Section 3.3.2, the constraints in Eqns.(14) and (15) are enforced by rejecting any switch that does not meet the criteria. After a switch is made, we check if the resulting plane positions adhere to all of the constraints. If so, the process continues and $\Delta W_{total,new}$ is calculated. If not, the switch is rejected and a new combination is found. These constraints are further enforced as in the pseudo code in Table 4.

Table 4: Extension of the pseudo-code in Table 3 for plane constraint enforcement

```
…
IF $u > g$ OR $\vec{p}_1 \cdot \vec{p}_2 \neq \vec{q}_1 \cdot \vec{q}_2$ OR $\vec{p}_3 \cdot \vec{p}_4 \neq \vec{q}_3 \cdot \vec{q}_4$ OR $(\vec{a}_1 - \vec{a}_2) \cdot \vec{p}_1 / |\vec{p}_1| \neq (\vec{b}_1 - \vec{b}_2) \cdot \vec{q}_1 / |\vec{q}_1|$
        $q_m = q_1^{(old)}$; $q_n = q_2^{(old)}$; $b_m = b_1^{(old)}$; $b_n = b_2^{(old)}$;
        $\Delta W_{total,new} = \Delta W_{total}$;
END
…
```

Plane-constrained structures, such as the graphite unit cell discussed in Section 3.2.2, have groups of atoms that remain in the same plane throughout the transition. The $\vec{p}_i$ that represents this plane in the initial state must transition to a specific $\vec{q}_i$ which represents that plane's location in the final state. If a switch is made that causes this plane to transition to a different location, that combination is rejected and a new switch is made.

After the correspondence of periodic surfaces between the initial and final states is found, interpolation of surfaces is used to find the locations of each atom for a number of intermediate states. Each atom moves along a transition path to its final position in a predetermined number of steps. Interpolation is only applied to the surfaces, yielding a new set of surfaces at each step. Each atom location at any stage is found by the intersection of three surfaces.

A simple surface linear interpolation approach is used to define the intermediate basis vectors between the initial and final vectors by linear interpolation between the two. Suppose that there are K stages during the morphing process. If a basis vector in the initial state (stage 1) is $\mathbf{p}_m^{(1)}$ and the corresponding one in the final state (stage K) is $\mathbf{p}_m^{(K)}$, then the basis vector $\mathbf{p}_m^{(k)}$ for the intermediate k th stage is given by

$$\mathbf{p}_m^{(k)} = (1 - l^{(k)})\mathbf{p}_m^{(1)} + l^{(k)}\mathbf{p}_m^{(K)} \quad (0 < l^{(k)} < 1 \text{ for } k = 2,\mathrm{K},K-1) \qquad (16)$$

with the interpolation coefficients $l^{(k)}$'s for all basis vectors $m = 1,\mathrm{K},M$. Particularly, $l^{(1)} = 0$ and $l^{(K)} = 1$. The intervals of interpolation coefficients $l^{(k)}$'s are chosen depending on how many intermediate states are desired.

The surface linear interpolation approach is a straight-forward method for finding intermediate steps in a phase transition process. The positions of atoms during the transition are nonlinearly interpolated between initial and final states by the surface linear interpolation. Yet, the physical interaction between atoms is not captured when deciding their intermediate positions. Physical forces may prevent atoms from colliding or getting too close to each other. To model the physical interaction, a second approach, potential driven surface interpolation, is also proposed here. The physical forces between atoms are captured by the potential between surfaces. Given two surfaces

$$y_i(\mathbf{r}) = \sum_{l_i=1}^{L_i}\sum_{m_i=1}^{M_i} m_{l_i,m_i} \cos\left(2pk_{l_i}(\mathbf{p}_{m_i}\cdot\mathbf{r})\right) = 0$$

and

$$y_j(\mathbf{r}) = \sum_{l_j=1}^{L_j}\sum_{m_j=1}^{M_j} m_{l_j,m_j} \cos\left(2pk_{l_j}(\mathbf{p}_{m_j}\cdot\mathbf{r})\right) = 0$$

the pair-wise potential between them is defined as

$$E(y_i,y_j) = \sum_{l_j<l_i}\sum_{m_j<m_i}\left[\exp\left(\left|m_{l_i,m_i} - m_{l_j,m_j}\right| + (a_{m_i}a_{m_j} + b_{m_i}b_{m_j} + c_{m_i}c_{m_j})\right)\cdot\cos^2\left(p(k_{l_i}+k_{l_j})(a_{m_i}-a_{m_j})\right)\right] \qquad (17)$$

Eqn.(17) combines the differences between the basis vectors and moments. Particularly, the lowest potential between two periodic planes that have the same normal direction is achieved when the distance in-between is the largest. That is, the two planes have a $p / 2$ phase difference. Two perpendicular planes also have a relatively low potential.

Suppose that there are a total of N surfaces in a model. The potential driven surface interpolation approach individually finds the interpolation coefficient $l_i^{(k)}$ for the i th surface at the k th stage, instead of the predetermined $l^{(k)}$'s for all surfaces. The process is to find $l_i^{(k)}$'s such that the total pair-wise potential for all surfaces at the k th stage is minimized. That is, we need to solve

$$\begin{aligned} &\min_{l_1^{(k)},\mathrm{K},l_N^{(k)}} \sum_{i=1}^{N} \sum_{j=i+1}^{N} E\left(y_i^{(k)}(l_i^{(k)}), y_j^{(k)}(l_j^{(k)})\right) \quad \mathit{for}\ k = 2,\mathrm{K}\ ,K \\ &s.t. \\ &(C1) \quad \textstyle\sum_{i=1}^{N} l_i^{(k)} = N l^{(k)} \\ &(C2) \quad l_i^{(1)} = 0 \text{ and } l_i^{(k-1)} - e \le l_i^{(k)} \le 1 \quad (i = 1,\mathrm{K}\ ,N) \end{aligned} \tag{18}$$

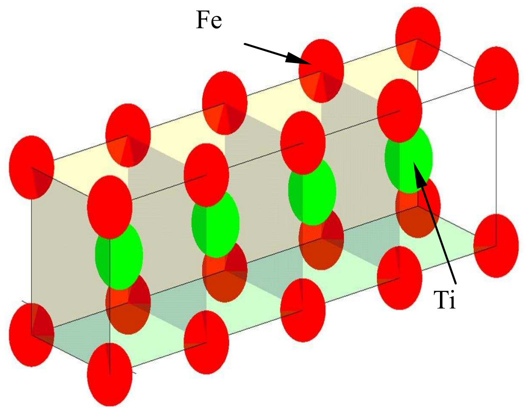
Notice that intermediate surface $y_i^{(k)}$ at stage k is a function of $l_i^{(k)}$ where its corresponding basis vectors are calculated similar to Eqn.(16). The equality constraint C1 is the boundary condition that the initial and final stages are met. At the same time, it ensures that the system evolves by stages with the predetermined values of $l^{(k)}$'s. The lower and upper bound constraint C2 ensures that the system evolves forward in general, where a small number e is introduced in the lower bound such that a limited setback is allowed to have more stable intermediate states with a lower potential level.

3.4 Results and Analysis

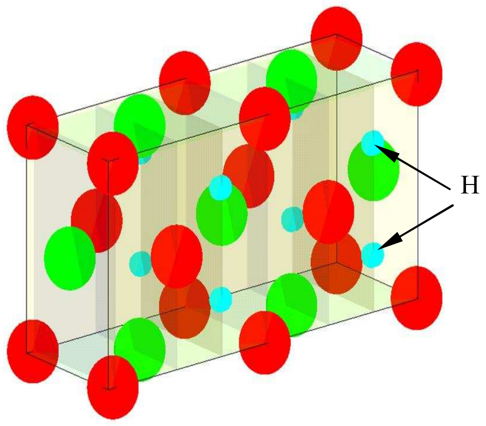
This section includes a demonstration of the loci-surface guided transition path search methods proposed in Chapter 3 by examples of iron-titanium (FeTi), vanadium dioxide (VO_2), and iron-platinum (FePt) phase transition. FeTi is being extensively studied as a candidate material for hydrogen storage applications. VO_2 thin films undergo changes during reversible and ultra-fast metal-to-semiconductor phase transition, which can be widely applied in high-volume rewritable holographic storage, high-speed fiber-optical switching, smart windows, etc. The layered state of FePt exhibits high magnetocrystalline anisotropy, making it potentially useful as a material in high density data storage. In order to search saddle points on the PES by methods such as the NEB, a good initial guess is required. The proposed geometry-guided path search method provides such an initial guess that is close to the minimum energy path.

3.4.1 FeTi+H Transition

FeTi experiences transition from a body-centric structure to an orthorhombic state where it can hold two hydrogen (H) atoms. Fig. 6-(a) shows four unit cells of the FeTi structure at its initial state. The unit cell of FeTi is body-centered cubic, where the Ti atoms are at the center and Fe atoms at the corners. The size of the unit cell is $a = b = c =$ 5.629 bohr [56]. Fig. 6-(b) shows one of the possible final states when two H atoms are absorbed in each unit cell forming the structure of FeTiH. This is an orthorhombic structure with Fe and Ti atoms on each face. Fe atoms still occupy the corners as well as the centers of the top and bottom faces. Ti atoms are located in the center of each side face. H atoms are located on two side faces. The size of the unit cell is a = 5.586 bohr, b 8.585 bohr, and c = 8.292 bohr [56]. Notice that Fig. 6-(b) shows two unit cells of FeTiH, which correspond to four unit cells of FeTi.



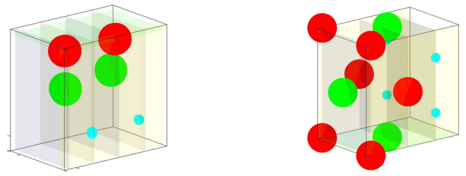
(a) FeTi



(b) FeTiH

Fig. 6: Comparison between FeTi and FeTiH

Geometry optimization or relaxation based on the ab initio molecular dynamics (CPMD) is performed first on both initial and final states of the FeTi+H transition using the software tool Quantum-Espresso [57]. Since searching the saddle point of the transition process, where H atoms are absorbed, requires us to have the same number of atoms in a unit cell, H atoms are introduced into the body-centered cubic FeTi structure to match the final FeTiH structure. In a physical experiment, the space around the FeTi material would be saturated with H atoms which are readily available for absorption. As shown in Fig. 7, there are two basis atoms for each type of Fe, Ti, and H in one unit cell of FeTiH as the final structure. Correspondingly, for two unit cells of body-centered FeTi, there are two Fe atoms and two Ti atoms as the basis of the initial structure, in addition to the two H atoms. In the initial structure, there is a H atom placed on the side of each unit cell, which is one of the most likely positions where H atoms are first absorbed in the cell [58]. The size of the unit cell for the initial structure is also set to be the size of the final structure, where meta-stable structure is likely to form. After the geometry optimization, meta-stable structures with the local minimum total energy are found, which are very close to the ones in Fig. 7.



(a) FeTi+H initial structure (b) FeTiH final structure

Fig. 7: Initial and final phases of FeTi with H absorbed

During the search of the initial transition path, atom locations for a unit cell of the two states in Fig. 7 are compared using the method described in Section 3.2. For each atom in the initial state, the corresponding location in the final state is found based on Eqn.(10). Three planes are defined for each atom in the initial or final state. The respective *y*-*z*, *x*-*z*, and *x*-*y* planes of atom i are

$$\begin{cases} \psi_i^{yz}\left(x\right) = \cos\left(\pi(x + a/2 + x_i)/2\right) = 0 \\ \psi_i^{xz}\left(y\right) = \cos\left(\pi(y + b/2 + y_i)/2\right) = 0 \\ \psi_i^{xy}\left(z\right) = \cos\left(\pi(z + c/2 + z_i)/2\right) = 0 \end{cases}$$

where a = 5.586, b = 8.585, and c = 8.292 define the size of the unit cell, and x_i, y_i, and z_i are the atoms' coordinates listed in Table 5.

Table 5: Initial and final geometris of FeTiH after relaxation (unit: bohr)

Initial structure				Final structure			
	x_i	y_i	z_i		x_i	y_i	z_i
Fe	0.000000000	0.000000000	0.000000000	**Fe**	0.000000000	-0.407605434	0.000000000
Fe	0.000000000	4.292490268	0.000000000	**Fe**	0.000000000	4.700081861	4.146037265
Ti	2.793000491	2.147048391	4.146037265	**Ti**	2.793000491	-0.104477173	4.146037265
Ti	2.793000491	6.437932145	4.146037265	**Ti**	2.793000491	4.396994224	0.000000000
H	0.000000000	0.000000000	4.146037265	**H**	0.000000000	2.146238663	2.072969363
H	0.000000000	4.292490268	4.146037265	**H**	0.000000000	2.146238663	6.219105168

Face correlation as described in Section 3.2.2 is used to reduce the number of loci planes. If we assume that the two basis Fe atoms are always on the same vertical y-z plane during the transition, the total number of planes is reduced from 18 to 17. Similarly, if the two Ti atoms and two H atoms are always on the same y-z plane respectively, the number of planes is further reduced to 15.

By the surface linear interpolation in Eqn.(16), the basis vectors of the planes that define the atom positions in the initial and final states, and the basis vectors for planes in the intermediate states can be found. The PS models in Fig. 8 represent six different states during the phase transition. Two unit cells of FeTi morph to one unit cell of FeTiH. It can be seen that the basis H atom on the right moves up while the basis H atom on the left moves down. At the same time, the basis Fe atom in the middle moves down towards the center of the face, while the basis Fe atom in the corner shifts right. For the two basis Ti atoms, the left one moves up to the top face while the right one moves further out of the unit cell.

Using the potential-driven surface interpolation in Eqn.(18), we receive a different initial guess of the transition path, as shown in Fig. 9. Compared to the previous one in Fig. 8, atoms tend to move individually one after another instead of simultaneously. Table 6 shows the detailed interpolation coefficients $l_i^{(k)}$'s for each plane at each stage as a result of minimizing potentials.

The initial guess of the transition path in Fig. 8 is imported as the input of the NEB method in Quantum-Espresso to find the MEP. The result is shown in Fig. 10, where each image at the bottom of the figure represents a state with a total of six. The initial and final states are the respective ones in Fig. 8, whereas the other four intermediate ones have been updated to reflect the MEP. The calculated energy level for each state is shown with the solid lines. Particularly, image 3, corresponding to −4713.9203 eV, has the highest energy level along the MEP. It is the saddle point found by the NEB method. The activation energy is 1.5771 eV, which corresponds to 0.26285 eV per atom. The second initial guess of transition path in Fig. 9 from the potential-driven surface interpolation is also used to run the NEB. The resulted MEP is shown as the dash lines in Fig. 10. The corresponding images are shown at the top of the figure. It is seen that the MEP found by the initial guess from the potential-driven surface interpolation gives a saddle point energy value of −4708.5716 eV. The activation energy found is 6.9258 eV, which corresponds to 1.1543 eV per atom. This is a higher energy level of the saddle point than the one from the surface linear interpolation. The lower energy saddle point reflects the true MEP better. In contrast, we also run the NEB method with its empirical default initial guess, which is the simple linear interpolation of atom positions. The result is also shown in Fig. 10, represented as the dotted lines. In this case, the NEB method fails in searching the saddle point after 100 NEB iterations, since there is no intermediate state that has higher energy level than both the initial and final states. Total CPU time required for the potential-driven method was 34 hours on a computer node with four CPU's, while the linear interpolation method required 14.5 hours. Experimentally, the activation energy for this material has been found to be 0.2912 eV per atom [59], which is very close to the result obtained with the surface linear interpolation method.

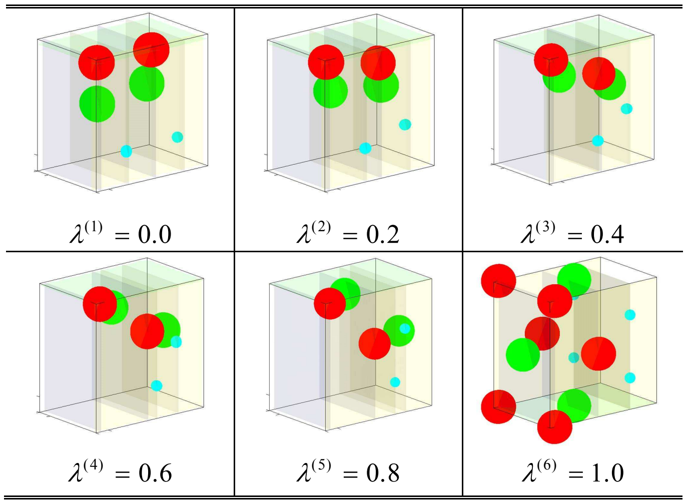


Fig. 8: Initial guess of the transition path for FeTi+H based on the surface linear interpolation in Eqn.(16)

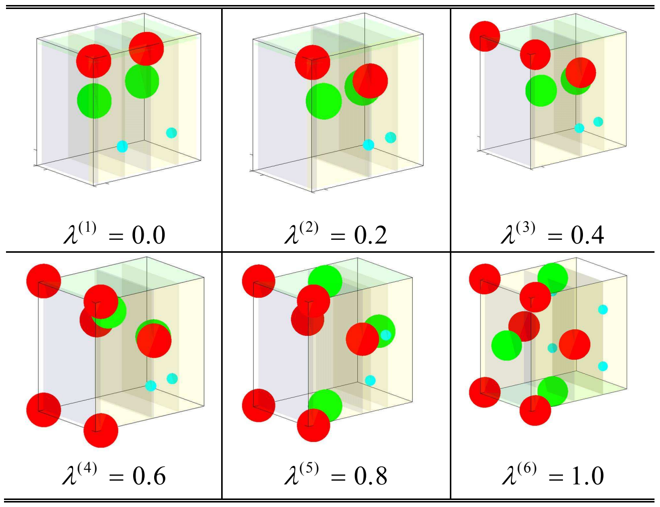


Fig. 9: Initial guess of the transition path for FeTi+H based on the potential-driven surface interpolation in Eqn.(18)

Table 6: Interpolation coefficients for y-z, x-z, x-y planes as the result of the potential-driven surface interpolation in Fig. 9

	k=1			*k*=2			*k*=3			*k*=4			*k*=5			*k*=6		
	y-z	x-z	x-y	y-z	x-z	x-y	y-z	x-z	x-y -	y-z	x-z	x-y	y-z	x-z	x-y	y-z	x-z	x-y
Fe	0	0	0	0	0	0	1	.9837	.6394	.9995	.9979	.9963	.9995	.9999	.9992	1	1	1
Fe	0	0	0	.6409	0	.4990	.6555	0	.5000	.9968	0	.7783	.9998	.1684	.7773	1	1	1
Ti	0	0	0	.0146	0	.4990	.3269	0	.4984	.9970	0	.9970	1	.6723	1	1	1	1
Ti	0	0	0	0	.4834	0	0	.4824	0	. 2289	.4814	.5361	1	.6723	.9994	1	1	1
H	0	0	0	0	.4990	0	0	.4980	0	0	.5361	0	.7328	.6723	.6723	1	1	1
H	0	0	0	.4990	.4912	0	.4980	.4902	.3269	.5361	.4892	.5361	.6723	.6723	.6723	1	1	1

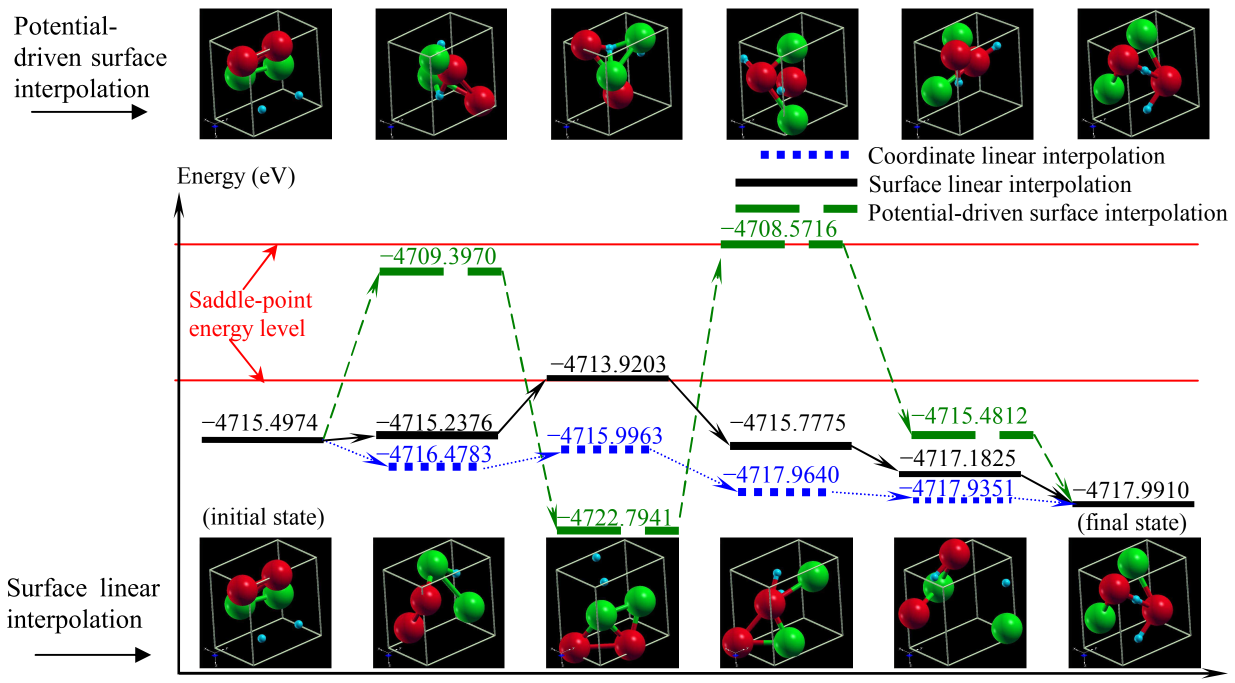


Fig. 10: Results of MEP in FeTi+H transition by the NEB method with different initial guesses

It can be seen from the results in Fig. 10 that both proposed methods are superior to the default empirical initial guess, which fails to find the saddle point in this case. It is also observed that our more involved, potential-driven surface interpolation method generates poorer results than the simpler linear interpolation method, as the resulting saddle point energy level is higher. One possible explanation for this result is that for the predicted intermediate states of the potential-driven interpolation method the atoms move individually, as seen in Fig. 9, rather than simultaneously as in Fig. 8. It has been discovered by first-principles simulations and experimentally observed (e.g. [60,61]) that the individual movement or single-hop diffusion sometimes requires a greater amount of energy than the coordinated diffusion. For this example, the linear interpolation method may provide a more accurate prediction of atom movement and therefore generate a better result in terms of saddle point energy levels. However, the potential driven surface interpolation method still provides more flexibility and a different guess of transition path; it also avoids paths which may result in atom collisions, which could be an important consideration.

3.4.2 VO_2 Transition

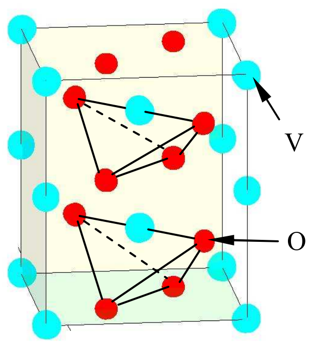
To demonstrate the process outlined in Section 3.3, we will use an example of VO_2 transition. The initial rutile phase and final M_2 phase of VO_2 are shown in Fig. 11. In each unit cell, there are eight oxygen (O) basis atoms. Following the procedure in Section 3.1, we build two tetrahedrons or eight different planes for each phase so that the positions of the O atoms can be uniquely determined by the intersections of the eight planes. Setting the values of c and s in Eqns.(11) and (12) to be at least 300, we can reliably find the correct correspondence of the periodic planes between the initial and final stages. The corresponding normal directions and the center points of the planes are listed in Table 7, where the minimum total potential change $\Delta W_{total} = 447.35$ is found.

Table 7: Initial and final locations and directions of planes for O in Fig. 11

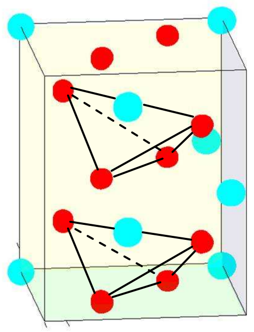
Plane	$\vec{a}_i$	$\vec{p}_i$	$\vec{b}_i$	$\vec{q}_i$
1	(3.4438,3.4438,0.8981)	[0.4201, 0.4201,0.8044]	(3.6265,3.5620,0.9862)	[0.4256,0.4265,0.7981]
2	(5.1631,5.1631,0.8981)	[-0.4201,-0.4201,0.8044]	(5.4706,5.4009,0.8778)	[-0.3957,-0.3965,0.8284]
3	(3.7285,4.8784,1.7961)	[-0.5242,0.5242,0.6712]	(3.9357,5.0874,1.8640)	[-0.5336,0.5719,0.6231]
4	(4.8784,3.7285,1.7961)	[0.5242,-0.5242,0.6712]	(5.0714,3.9539,1.8640)	[0.5382,-0.4997,0.6787]
5	(3.4438,3.4438,6.2864)	[0.4201, 0.4201,0.8044]	(3.7540,3.5504,6.5575)	[0.3982,0.3865,0.8319]
6	(5.1631,5.1631,6.2864)	[-0.4201,-0.4201,0.8044]	(5.5981,5.3893,6.6658)	[-0.4493,-0.4361,0.7797]
7	(3.7285,4.8784,7.1844)	[-0.5242,0.5242,0.6712]	(4.0220,5.0766,7.5656)	[-0.5939,0.5616,0.5761]
8	(4.8784,3.7285,7.1844)	[0.5242,-0.5242,0.6712]	(5.1126,3.9530,7.5656)	[0.4960,-0.5373,0.6820]

Similar to the previous example in Section 3.4.1, the surface linear interpolation method is used to find the initial guess of transition path. After the NEB search, the respective energy levels for seven images are: 1) −5006.1158eV, 2) −5000.6632eV, 3) −5003.8746eV, 4) −4997.6627eV, 5) −4990.0234eV, 6) −4989.7258eV, and 7) −5006.4782eV. Image 6) has the highest energy and therefore represents the saddle point.

The activation energy found is 16.39 eV, which corresponds to 1.37 eV per atom. The experimental data for the activation energy is approximately 0.6 eV per atom [62]. Using the empirical default initial guess, the NEB fails to locate the saddle point again.



(a) VO_2 initial rutile phase

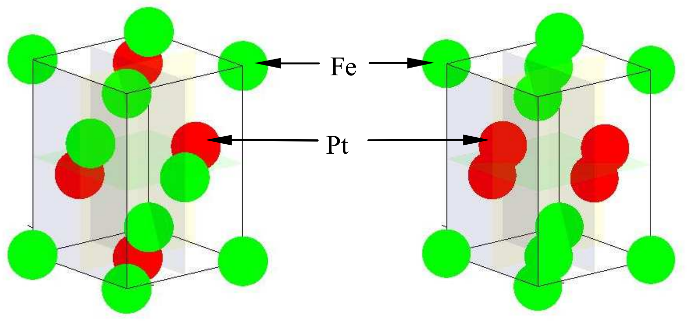


(b) VO_2 final M_2 phase

Fig. 11: Initial and final phases of VO_2

3.4.3 FePt Transition

The unit cell of the initial disordered A1 state of FePt is face centered cubic with two iron (Fe) and two platinum (Pt) atoms each. The structure transitions into a layered $L1_0$ face centered tetragonal phase, where atoms of the same species are located in the same plane. Both phases are shown in Fig. 12. In the final phase, the dimensions of the unit cell are a=3.874 bohr and c=3.714 bohr [63]. Similar to the previous examples, the surface linear interpolation is used to generate an initial guess of atom locations during the transition, where the activation energy found is 0.8099 eV per atom. The result from the empirical default initial guess is 0.7602 eV per atom. Both are reasonably close to the experimentally measured 1.7 eV per atom [64].



(a) FePt initial A1 phase (b) FePt final $L1_0$ phase

Fig. 12: Initial and final phases of FePt

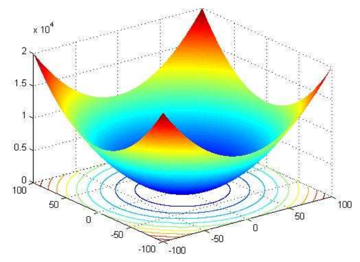
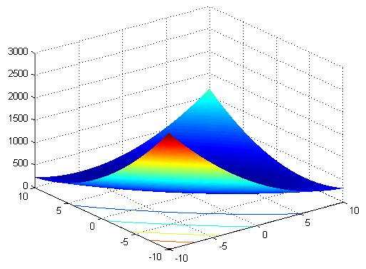
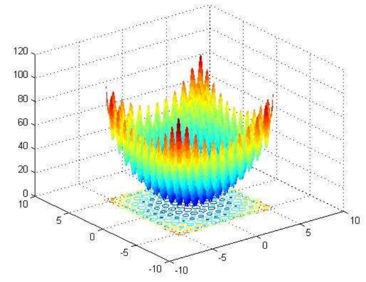
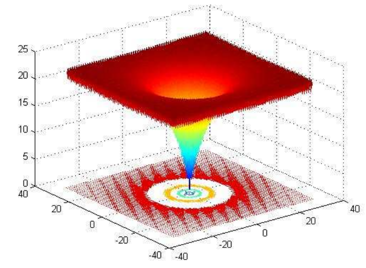
4. PARTICLE SWARM OPTIMIZATION FOR STRUCTURE RELAXATION

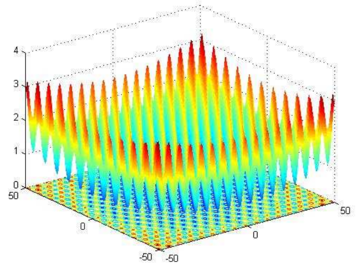
As discussed in section 2.4, the original PSO method is inspired by the movement of a group of insects or birds who fly collectively as a group and also learn from each other about potential sources of food. Using this type of behavior, the swarm can search a larger area and at the same time relay useful information to all the individuals in the group. Similarly, an optimization algorithm based on this principle is useful for finding the global minimum or maximum of a numerical function on a continuous domain. To address some of the drawbacks of the original PSO algorithm, two improved methods are proposed in the following sections, the "Random Group" PSO in section 4.1, and the "Active Group" PSO in section 4.2. Then, experiments that were done with each method are presented in section 4.3, and finally the results of the experiments and performance evaluation are presented in section 4.4.

The default method used within Quantum Espresso for relaxation is a BFGS algorithm, as discussed in section 2.4. This algorithm is a local optimization method that searches the space near the given start and end positions for a local optimum. Usually a local optimum is found very close to the starting positions, so that the relaxed crystal structure coordinates are very similar to the input coordinates. However, a relaxed position with a lower potential energy may be available, which would be more favorable for transition path search. In order to find other optima in the space, and possibly arrangements with lower potential energy, a global optimization method is required. Using the standard PSO algorithm, we can investigate the search space more thoroughly, and increase the chance of finding a more relaxed position. Additionally, as will be shown in the following sections, the proposed methods allow us to search the space for a global optimum as well as several local optima using additional groups. By replacing the native BFGS method with PSO algorithms, we are trying to improve the simulation by finding a lower energy relaxed position for the initial and final structures.

To investigate the robustness and efficiency of the PSO algorithm compared to the two proposed methods, each algorithm is applied to the five test functions shown in Table 8. These functions are commonly used in the field of continuous function optimization and provide sufficient variety to test for robustness. The Booth and Sphere functions are unimodal; they have a single local optimum that is also the global optimum. The others are multimodal, i.e. they have multiple local optima and one global optimum. All of the functions have a global optimum value where $f(x) = 0$.

Table 8: Test functions

Sphere		$f(x)=\sum_{i=1}^{n} x_i^2$
Booth		$f(x)=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$
Rastrigin		$f(x)=10n+\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i))$
Ackley		$f(x)=-20\cdot\exp(-0.2\cdot\sqrt{1/n\cdot\sum_{i=1}^{n}x_i^2})-$ $\exp(\frac{1}{n}\cdot\sum_{i=1}^{n}\cos(2\pi\cdot x_i))+20+e$

Griewank		$f(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$

4.1 Random Group PSO

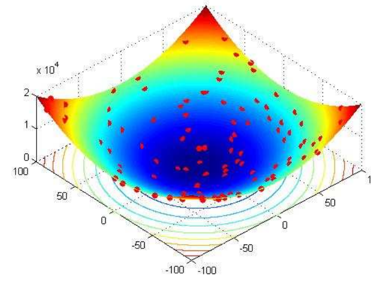
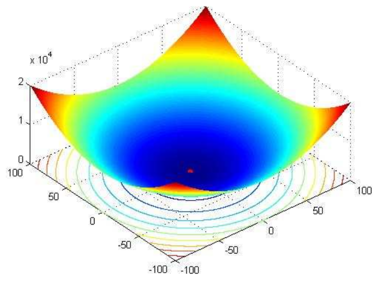
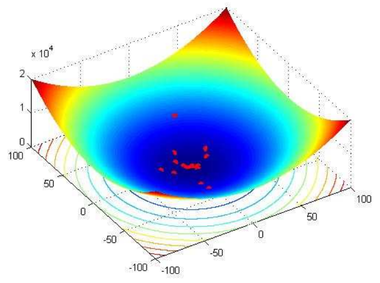
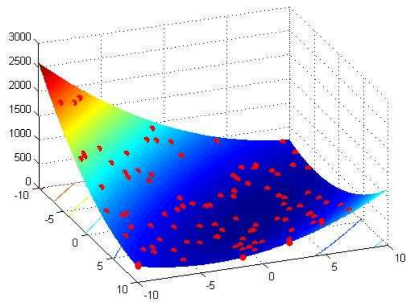
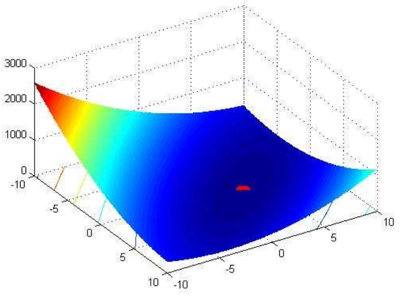
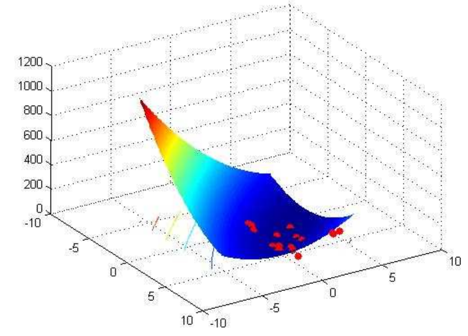
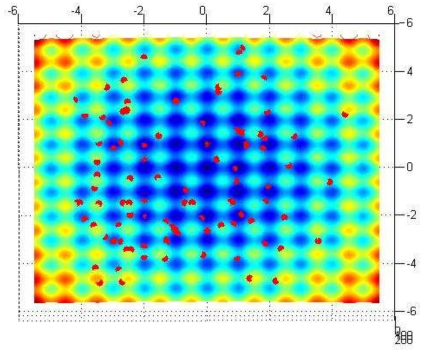
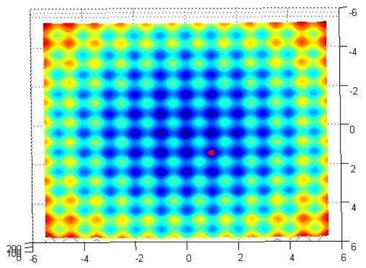
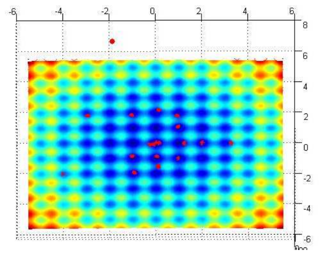
To increase the chance of finding the global optimum and provide information about local optima, the Random Group PSO (RG-PSO) divides the initial population of particles into groups of size n. Each group is also assigned an initial criticality value of zero. The updates then proceed similar to the standard PSO, but only particles from the same group are considered as neighbors. At each iteration, the position of each group is calculated as the average position of its' particles. Then, a Euclidean distance calculation is made to find the distance between groups. If the average position of particles in any group moves within the threshold distance (td) of the average position of any other group, the criticality value for one of the groups is increased by one. If any group's criticality reaches a predetermined "maxcrit" value or higher, all of its particles are relocated randomly in the search space. This re-location scheme allows only one group to converge to a particular location in the search space, regardless of whether the location is a local or global minimum. At the same time, given a sufficient number of iterations, the remaining groups have a chance to converge to another minimum. The algorithm increases the probability of finding the global optimum by avoiding premature convergence to local optima, and also provides more information about the landscape as different groups converge to various local optima. Table 9 shows a comparison of results using the original PSO method with results using RG-PSO and the AG-PSO method discussed in the next section. Visually, the two proposed methods have similar results, so a single picture is used to represent both methods simultaneously. The red and blue markers represent the average location of each group for RG-PSO and locations of each particle for regular PSO. It is obvious that the proposed methods are not ideal for the Booth and Sphere functions, or unimodal functions in general, as the different groups are unable to converge to the global optimum and simply end up distributed near the global optimum location. With the Rastrigin function as well as the other multimodal functions, however, we can see several groups converging on the global optimum in the center, as well as other groups converging to the various local minima. By increasing the number of particles and groups in the population, we can also increase the number of local minima that will be found. At the same time, it can be seen that the particles in the standard PSO tend to converge to a single location, which is oftentimes not the global optimum.

For testing purposes the threshold distance was set to 0.5 percent of the search space, and a maxcrit value of four was used. The velocity updates are now given by

$$V_j^{(i)} = \vartheta V_j^{(i-1)} + c_1 r_1 [P_{best,j} - X_j^{(i-1)}] + c_2 r_2 [R_{best} - X_j^{(i-1)}]$$

where R_{best} is the best previous position of any particle in the group. A pseudo-algorithm for the RG-PSO process is shown in Table 10.

Table 9: Particle distribution comparison between original PSO and proposed techniques

Function	Initial Distribution	Final Distribution (PSO)	Final Distribution (RG-PSO/AG-PSO)
Sphere			
Booth			
Rastrigin			

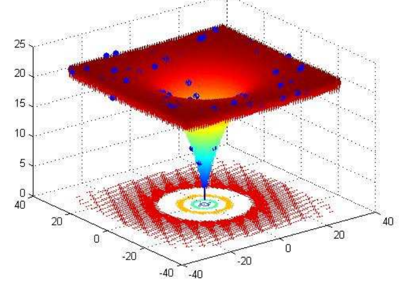
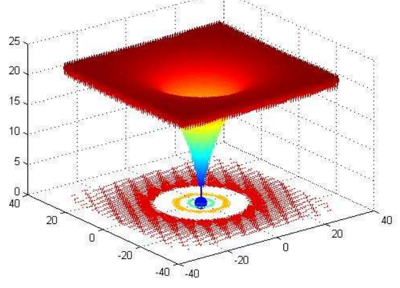
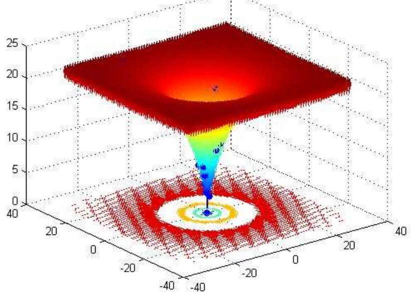
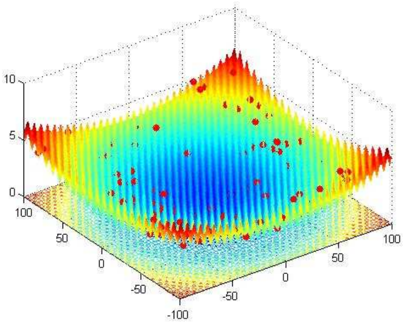
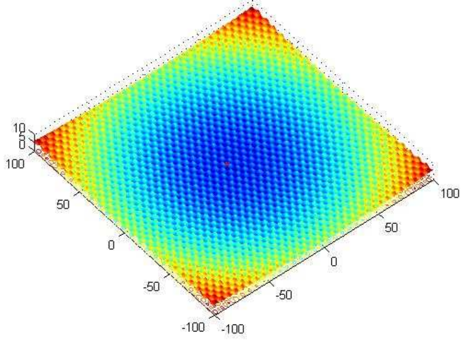
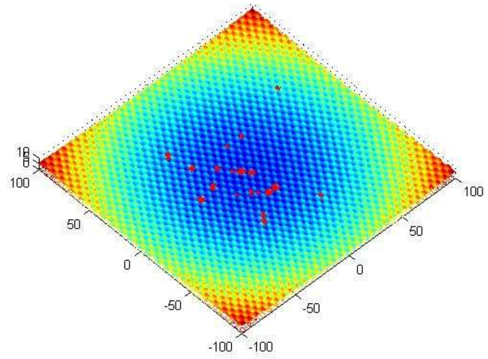
Ackley			
Griewank			

Table 10: Pseudo-code for the RG-PSO method

INPUT: number of particles $\{N\}$, number of particles in each sub-group $\{gsize\}$, upper and lower search space boundaries $\{b_u, b_l\}$, objective function $\{f(x_1,...,x_N)\}$, maximum number of iterations $\{Maxi\}$, constants for local and global directions $\{c_1, c_2\}$, movement restriction $\{J\}$, convergence threshold $\{epsilon\}$, threshold distance $\{td\}$, maximum allowed criticality before relocation $\{mcrit\}$

OUTPUT: location of global optimum $\{x_1,...,x_N\}$

```
FOR $i = 1:N$
    $X_i = rand()*(b_u - b_l) + b_l$
    $pbest_i = X_i$
    $V_i = 0$
    $f_i = f(X_i)$
END
$groups = N / gsize$
FOR $i = 1:groups$
    $gbest_i = \arg\min(f_{(i-1)*gsize+1}, f_{(i-1)*gsize+2}, \dots, f_{i*gsize})$
    $fgbest_i = \min(f_{(i-1)*gsize+1}, f_{(i-1)*gsize+2}, \dots, f_{i*gsize})$
END
FOR $i = 1:Maxi$
    FOR $j = 1:groups$
        FOR $n = (j*gsize - gsize + 1):(j*gsize)$
            $V_n = \vartheta * V_n + c_1 * rand()*(pbest_n - x_n) + c_2 * rand()*(gbest_j - x_n)$
            $X_n = X_n + V_n$
        END
        $avglocation_j = (sum(x((j*gsize - gsize + 1):(j*gsize)))) / gsize$
    END
    FOR $j = 1:groups$
        FOR $k = (j+1):groups$
            $dist_j = sqrt((avglocation(k) - avglocation(j))^2)$
            IF $dist_j < td$
                $crit_k = crit_k + 1$
                IF $crit_k > mcrit$
                    FOR $h = (k*gsize - gsize + 1):(k*gsize)$
                        $X_h = rand()*(b_u - b_l) + b_l$
                        $V_h = 0$
                    END
                END
            END
        END
        END
        FOR $n = 1 : N$
            $newf_n = f(X_n)$
        END
        FOR $a = 1 : groups$
            FOR $k = ((a * gsize) - gsize + 1) : (a * gsize)$
                IF $newf_k < fgbest_a$
                    $gbest_a = X_k$
                    $fgbest_a = newf_k$
                END
            END
        END
        FOR $n = 1 : N$
            IF $newf_n < f_n$
                $pbest_n = X_n$
            END
        END
        $f = newf$
        IF $\min(f_i) < epsilon$
            BREAK
        END
END
```

4.2 *Active Group PSO*

The Active Group PSO (AG-PSO) algorithm is similar to the RG-PSO in that the initial population is divided into smaller groups. However, rather than re-distributing particles randomly when two groups get too close to each other, each group actively moves away from its' m nearest neighboring groups. The velocity update is

$$V_j^{(i)} = \vartheta V_j^{(i-1)} + c_1 r_1 [P_{best,j} - X_j^{(i-1)}] + c_2 r_2 [G_{best} - X_j^{(i-1)}] - (c_3 \sum_{k=1}^{m} r_{k+2} [G_{k,best} - X_j^{(i-1)}])$$

where $G_{k,best}$ is the best location of the nearest neighboring group *k*, c_1, c_2, and c_3 are constants, and r_k $(k = 1, \dots, m+2)$ is a random number between 0 and 1. A pseudo-algorithm of the AG-PSO process is shown in Table 11.

Like the RG-PSO, the AG-PSO improves the probability of finding a global optimum when compared to standard PSO and also provides more information about the landscape. Initial and final particle distributions are similar to those for RG-PSO seen in Table 9. The intent with the AG-PSO is to avoid wasteful iterations where two or more groups move towards each other before being randomly re-distributed. By actively moving away from each other, the groups are less likely to stay in the same area for multiple iterations. The algorithm should become more efficient as it no longer has to wait to reach the threshold distance and maximum criticality before exploring a different region. By adjusting the value of c_3, AG-PSO can also be more effective at exploring regions that are densely populated with local optima. As two groups approach an area with multiple local optima, they may search within threshold distance of each other if the optima are sufficiently close. Rather than being re-distributed as with RG-PSO, the two groups may converge to different optima within a small region. A similarly thorough search with RG-PSO would require a smaller threshold distance, which makes the algorithm less efficient.

Table 11: Pseudo-code for the AG-PSO method

INPUT: number of particles $\{N\}$, number of particles in each sub-group $\{gsize\}$, upper and lower search space boundaries $\{b_u, b_l\}$, objective function $\{f(x_1,\ldots,x_N)\}$, maximum number of iterations $\{Maxi\}$, constants for local and global directions $\{c_1,c_2\}$, constants for movement away from neighboring groups $\{c_k\}$,movement restriction $\{J\}$, convergence threshold $\{epsilon\}$, number of neighboring groups to be considered $\{m\}$

OUTPUT: location of global optimum $\{x_1,\ldots,x_N\}$

```
FOR $i = 1:N$
    $X_i = rand()*(b_u - b_l) + b_l$
    $pbest_i = X_i$
    $V_i = 0$
    $f_i = f(X_i)$
END
$groups = N / gsize$
FOR $i = 1:groups$
    $gbest_i = \arg\min(f_{(i-1)*gsize+1}, f_{(i-1)*gsize+2}, \ldots, f_{i*gsize})$
    $fgbest_i = \min(f_{(i-1)*gsize+1}, f_{(i-1)*gsize+2}, \ldots, f_{i*gsize})$
END
FOR $i = 1:Maxi$
    FOR $j = 1:groups$
        $avglocation_j = (sum(x((j*gsize - gsize + 1):(j*gsize)))) / gsize$
    END
    FOR $j = 1:groups$
        FOR $k = 1:groups$
            $dist_{j,k} = |avglocation(k) - avglocation(j)|$
        END
    END
    FOR $j = 1:2:(groups*2)$
        $M_{j,:} = dist_{j/2}$
        $M_{j+1,:} = 1:groups$
        $B_{:,j:j+1} = sortrows(M_{j:j+1,:})$
    END
    FOR $j = 1:groups$
        FOR $n = (j*gsize - gsize + 1):(j*gsize)$
            $vadd(n) = zeros$
                FOR $k = 2:(neighbors + 1)$
                    $vadd_n = vadd_n + (c_3 * rand() * (avglocation_{B_{k,j*2}} - X_n))$
                END
            $V_n = \vartheta * V_n + c_1 * rand() * (pbest_n - x_n) + c_2 * rand() * (gbest_j - x_n) - vadd_n \backslash$
            $X_n = X_n + V_n$
        END
    END
    FOR $n = 1 : N$
        $newf_n = f(X_n)$
     END
    FOR $a = 1 : groups$
        FOR $k = ((a * gsize) - gsize + 1) : (a * gsize)$
            IF $newf_k < fgbest_a$
                $gbest_a = X_k$
                $fgbest_a = newf_k$
            END
        END
    END
    FOR $n = 1 : N$
        IF $newf_n < f_n$
            $pbest_n = X_n$
        END
    END
    $f = newf$
    IF $\min(f_i) < epsilon$
        BREAK
    END
END
```

Both the RG-PSO and AG-PSO methods involve the calculation of Euclidean distances at each iteration; as a result the computation time per iteration is increased compared to the standard PSO algorithm. This increase is offset by the increased accuracy and fewer iterations required to find the global optimum. Further discussion about the performance of PSO, RG-PSO and AG-PSO follows in section 4.4.

4.3 Experiments

In all of the experiments, the so called *gbest* PSO algorithm is used, where the entire population of particles is considered to be the neighborhood, and G_{best} is the best particle in the entire population. Parameter values $c_1 = c_2 = 1.7$ and $\vartheta = 0.6$ are used as recommended in [65]. Standard PSO, RG-PSO, and AG-PSO are each run 100 times for each test function, and the average results are presented in Table 13. Avg, Med, Max, and Min represent the Average, Median, Maximum, and Minimum number of iterations, respectively, needed to find the global optimum. Suc is the success rate of the algorithm; that is, how often it converged to the global optimum. And finally, Avg(s) is the average time, in seconds, that was needed to complete each run. Failed runs where the maximum number of iterations is reached were not included for the calculation of Avg, Med, Max, and Avg(s).

In each case, the particles have been initialized with a random position with the values randomly chosen within the minimum and maximum boundaries depending on the objective function. However, the position and velocity during subsequent iterations are not restricted to within these boundaries. Table 12 shows the boundary values for the range of possible initial positions, the objective function evaluation goals to be achieved by the algorithms, as well as the dimension that was used for each test function. The maximum number of iterations is set at 1000 in each case, and a swarm size of $N = 100$ was used with a group size of 5 in the case of RG-PSO and AG-PSO. Dimensions larger than two are possible and generally make the optimization more difficult. Because the application in this case is to structure relaxation, where the input objective function that describes the potential energy surface oftentimes has a dimension larger than two, test functions with higher dimensions are considered in section 4.5. For now, however, only two dimensional test functions are used, as these allow for relatively fast computation times and are sufficient to test the accuracy and robustness of the algorithms.

For the RG-PSO algorithm, the velocity of the particles in a group is reset to zero if the group is relocated. The threshold distance is set as 1% of the upper search space boundary. Additionally, for RG-PSO as well as AG-PSO, the initial ϑ value is set to 0.6, and is then decreased by 0.1 every 200 iterations. This gradual decrease of the ϑ value was not implemented with the standard PSO algorithm, as it did not require more than 200 iterations for most of the cycles. The number of neighbors, m, was set to 3 for AG-PSO, and a c_3 value of 0.3 was used for the Ackley function, while $c_3 = 0.5$ was used for the Sphere, Booth, Rastrigin, and Griewank functions. For the Ackley function, AG-PSO with c_3 greater than 0.3 produced very low success rates and the algorithm was not competitive, while the other functions did not show significant improvement when c_3 was lowered.

Table 12: Parameters for Test Functions

Name	Initial Position Boundaries	Goal	Dimension
Sphere	$[-100;100]^2$	0.01	2
Booth	$[-10;10]^2$	0.01	2
Rastrigin	$[-5.5;5.5]^2$	0.1	2
Ackley	$[-32;32]^2$	0.1	2
Griewank	$[-600;600]^2$	0.1	2

4.4 *Results and Significance*

The RG-PSO and AG-PSO algorithms were expected to take more time per iteration because a Euclidean distance calculation is involved that is not required with the standard PSO. Indeed, this is the case for the multimodal functions. Table 13 shows that both RG-PSO and AG-PSO have a large Avg(s) for the Rastrigin, Ackley, and Griewank functions. However, for the unimodal functions, RG-PSO outperformed standard PSO in completion time as well as average iterations. This may be due to the reduced number of neighbor comparisons with the RG-PSO. Unlike the standard PSO, the RG-PSO algorithm does not have to compare each particle to all others in the population. Instead, each particle is only evaluated with respect to the four others in its' group, and the groups are compared to each other, reducing the overall number of computations that are required. Additionally, thanks to the fast convergence with unimodal functions, there is not as much time to reach maximum criticality, so the additional steps of checking threshold distance and re-arranging individual groups are not as frequent as they are with multimodal functions.

AG-PSO did not perform well for any of the cases; completion time and average iterations are significantly higher. The algorithm performs better when c_3 is reduced, and with c_3 close to zero the results are similar to those of the RG-PSO (Avg=12.53, Avg(s)=0.0616 with c_3=0.1). When c_3 is significantly reduced or even set equal to zero, AG-PSO becomes similar to RG-PSO, without the additional step of re-positioning groups when they reach maximum criticality. In fact, it is also similar to the standard PSO in this case, so we should expect the results with very low c_3 values to be comparable to those obtained with PSO and RG-PSO. However, the intent with the AG-PSO is to prevent groups from clumping up at local or even global optima, and this effect is lost when c_3 is zero. Therefore, a c_3 value of zero is not advisable, and values between 0.3 and 0.5 were used in the experiments. In general, a more active movement away from the nearest neighbors using higher values of c_3 similar to c_1 and c_2 proved to be highly ineffective, with inaccurate results and long computation times. Because the implementation of c_3 and the movement away from other particles counteracts the base PSO algorithm's tendency to continuously move the particles towards each other, c_3 values that are similar to c_1 and c_2 cause a situation where groups and particles are simultaneously drawn in opposite directions. This results in many iterations where individual groups or particles move very little from their previous positions, and subsequently the efficiency of the algorithm is reduced. Values of c_3 that are significantly higher than c_1 and c_2 allow for faster convergence but lower accuracy. Particles are allowed to move relatively quickly, but the region near the global optimum is not searched as thoroughly as with standard PSO.

The RG-PSO algorithm, on the other hand, showed very good performance across all example functions. As shown in Table 13, the success rate is 1.00 for each function, and average cycle completion times increase consistently with function complexity. In some cases, such as with the Rastrigin function, RG-PSO yielded success rates of 100% even when the convergence criteria was narrowed from 0.1 to 0.01. Overall completion times are similar to or lower than those for the standard PSO algorithm on all but the Rastrigin function, where PSO showed fast convergence but very low success rates. Overall, the RG-PSO algorithm displayed the best combination of accuracy and fast convergence, providing consistently high success rates as well as information about local optima if required.

The success rate for PSO was for the most part consistent with previously obtained results that can be seen in the literature. With the Rastrigin function, however, it is unusually low. Even using a convergence goal of 0.1 rather than the 0.01 that was used with RG-PSO, the PSO algorithm showed very low success rates. Possibly due to the geometry of this function, approximately 40% of the time the PSO algorithm settles in a local optimum and is unable to escape, and with the convergence goal set at 0.01 the success rate drops even lower to approximately 36%. As a result, the global optimum is not found within the iteration limit. One reason the standard PSO is significantly less successful on the Rastrigin function than on the other multimodal functions may be because the function values at the local optima are very close to those at the global optimum. Unlike the other multimodal functions used here, this allows the algorithm to converge to a location that is believed to be the global optimum, as it meets the 0.1 convergence criteria. This allows the algorithm to quickly settle in a location that meets the convergence criteria but is not the global optimum. It is shown, however, that the use of sub-groups with RG-PSO and AG-PSO results in a success rate close to 1.00 for all of the test functions, giving them an advantage over standard PSO.

Table 13: Iterations Required to Achieve Goals for Each Algorithm and Test Function

Algorithm		**Avg**	**Med**	**Max**	**Min**	**Suc**	**Avg(s)**
		Sphere					
PSO		28.9	28	66	9	1.00	**0.138**
RG-PSO		26.3	25	49	2	1.00	**0.0681**
AG-PSO		478	486	641	23	1.00	**2.47**
		Booth					
PSO		39.5	16.5	567	1	0.94	**0.429**
RG-PSO		12.3	12	21	1	1.00	**0.0539**
AG-PSO		232	194.5	574	2	1.00	**1.12**
		Rastrigin					
PSO		21.5	17	80	1	0.59	**0.102**
RG-PSO		84.6	58	384	9	1.00	**0.253**
AG-PSO		482	494.5	943	3	0.96	**2.15**
		Ackley					
PSO		30.5	29	79	11	1.00	**0.0752**
RG-PSO		29.9	28	90	6	1.00	**0.143**
AG-PSO		44.6	43.5	104	16	1.00	**0.216**
		Griewank					
PSO		18.7	16.5	55	3	1.00	**0.0706**
RG-PSO		18.4	17	53	1	1.00	**0.1007**
AG-PSO		359	449	633	13	1.00	**1.72**

4.5 PSO Integration With Quantum Espresso

In this section, the methods for integrating the original and proposed PSO algorithms within QE in place of the native BFGS algorithm are discussed. A pseudo-algorithm for the BFGS method discussed in section 2.4 is shown in Table 14. This BFGS algorithm iteratively updates each atom's position until an acceptable overall minimum energy location is found. Although this approach has proven to be robust within the QE software, it may be possible to increase the efficiency of the process by replacing the existing BFGS method with a PSO algorithm. Rather than using one set of atom locations and updating each atom's position individually, we can use each particle in the PSO method to represent one possible set of atom locations. Therefore, it becomes possible to simultaneously search a larger region in the search space and reduce the amount of time required to find the relaxed position.

Table 14: Pseudo-code for standard BFGS algorithm

```
INPUT: set of $k$ locations as initial guess $\{\vec{x}_0\}$, objective function $\{f(\mathbf{x}_i)\}$, gradient of the objective function $\{\vec{g}_i\}$, initial approximate Hessian matrix $\{B_0\}$, initial step size $\{a_0\}$, convergence threshold $\{epsilon\}$
OUTPUT: set of $i$ optimized locations $\{\vec{x}_i\}$
WHILE $(f(\vec{x}_{i+1}) - f(\vec{x}_i)) > epsilon$
    FOR $i = 1:k$
        $\vec{p}_i = -B_i^{-1}\vec{g}_i$
        $\alpha_i = \vec{x}_{i+1} - \vec{x}_i$
        $\vec{S}_i = \alpha_i \vec{p}_i$
        $\vec{y}_i = \vec{g}_{i+1} - \vec{g}_i$
        $B_{i+1}^{-1} = \frac{B_i^{-1} + (\vec{S}_i^T \vec{y}_i + \vec{y}_i^T B_i^{-1} \vec{y}_i)(\vec{S}_i \vec{S}_i^T)}{(\vec{S}_i^T \vec{y}_i)^2} - \frac{B_i^{-1} \vec{y}_i \vec{S}_i^T + \vec{S}_i \vec{y}_i^T B_i^{-1}}{\vec{S}_i^T \vec{y}_i}$
        $\vec{x}_{i+1} = \vec{x}_i + \alpha_i p_i$
    END
END
```

Comparing the standard BFGS algorithm shown in Table 14 to the version that is implemented in QE, there is one notable difference. The standard BFGS process uses a "WHILE" loop to continue making position updates until a satisfactory objective function value is reached. This means that when the algorithm is started, it will run either until the condition is satisfied, or a maximum allowable number of iterations is reached. On the other hand, the BFGS algorithm module implemented in QE omits the “WHILE” loop. Instead, only one iteration and position update is done each time the algorithm is called, and the loop and exit criteria are stored in a separate file. Most pertinent data, such as initial atom locations, gradients, and the energy evaluation subroutine are stored in a separate directory as well and called each time the BFGS algorithm is used. At the end of the iteration, the updated information is written back into the file, and called again when the next iteration begins. The criteria for termination of the BFGS process is stored and evaluated outside of this BFGS module, and the algorithm is called repeatedly until the convergence criteria is met. This difference must be considered when implementing the various PSO methods shown in Table 2, Table 10, and Table 11 because multiple iterations, rather than one, are computed when the algorithm is called. Additionally, the objective function must be evaluated for each particle in the PSO algorithm, so multiple function calls are required at each iteration rather than the single evaluation that is needed for each iteration of BFGS.

Considering these changes, the PSO algorithm was implemented to work seamlessly with the existing module. Since the energy calculation is performed in a separate file, the algorithm is a modified version of that shown in Table 2. The pseudo-code for the PSO method implemented for relaxation is shown in Table 19. Rather than evaluating an objective function for each particle, the “electrons()” subroutine is called to calculate total energy based on atom positions, and the resultant total energy is returned as “etot”. Each particle within the PSO represents a possible arrangement of atom positions; therefore the subroutine must be called once for each particle within each iteration. In addition to these changes, the code for each of the PSO algorithms had to be translated from its MATLAB version into FORTRAN code that is used in QE. A copy of these algorithms is provided in the Appendix.

The results for atom positions and final energy are shown in Table 16, and computation times for relaxation of the initial and final states are shown in Table 15. A PSO algorithm with ten particles was used, and 15 particles with five groups were used in the RG-PSO and AG-PSO algorithms. When applied to relaxation, the standard PSO algorithm requires a longer computation time than the native BFGS algorithm. This is to be expected because the electrons() subroutine is called ten times within each iteration, compared to just a few calls overall for the BFGS algorithm. It can also be seen from Table 16 that the atom positions did not match those that were obtained using BFGS, but in some cases the minimum energy that is found is lower. For the examples shown, it is likely that many local minima exist, and the BFGS method simply converges to a location close to the starting position. The global search within the PSO method, however, can find the global optimum or a local optimum that is different from those found with BFGS. Therefore, it is not likely that the same set of atom locations will be found. It is also seen in Table 15 that computation times for RG-PSO and AG-PSO were significantly lower than those for PSO, and in some cases also lower than BFGS. This may be due to the low number number of groups that were used; each group only had to be compared to four other groups.

To visualize the energy surface for the FeTiH and VO_2 structures, contour plots for the total potential energy of the structures are shown in Fig. 13 and Fig. 14, respectively. These plots were generated by keeping all but two of the dimensions constant, and varying the remaining two across a range of values. For the FeTiH structure, the y and z coordinates for one of the H atoms were varied, and similarly the y and z coordinates were varied for one of the O atoms in the VO_2 structure. The initial state was used as the default configuration for each structure, and each of the two coordinates were adjusted between 0.5 and 5 in increments of 0.5, resulting in a total of 100 data points. The unrelaxed positions for each atom can be seen in appendix sections 6.7 and 6.8. At each set of values, the total potential energy of the structure was evaluated using a QE module and plotted with Matlab, with the y-coordinate shown on the x-axis, and the z-coordinate on the y-axis. This approach was used because the structures are relatively complex, with twelve atoms and 36 dimensions that can be adjusted in the VO_2 example. A plot of the PES that involves variation of all 36 parameters would have been very involved and difficult to show on a 2- or 3-dimensional graph, so this simplified approach was used. However, even with this simplified plotting method and relatively small range of values, we can see that the surface has many local minima, making it a challenging problem for global optimization.

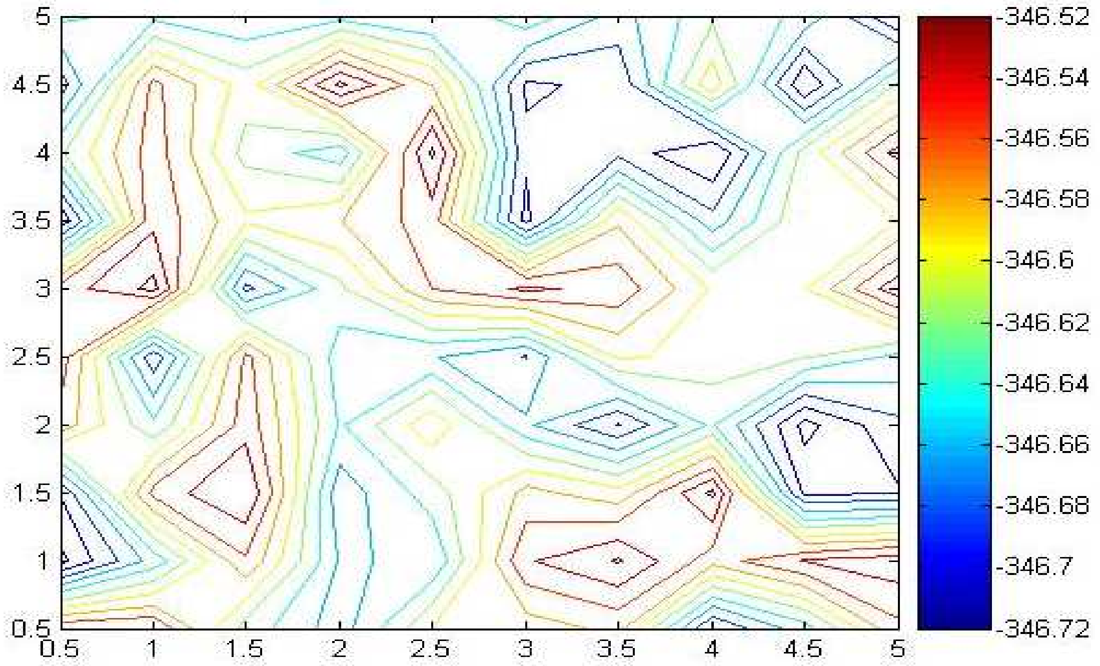


Fig. 13: FeTiH energy contour plot

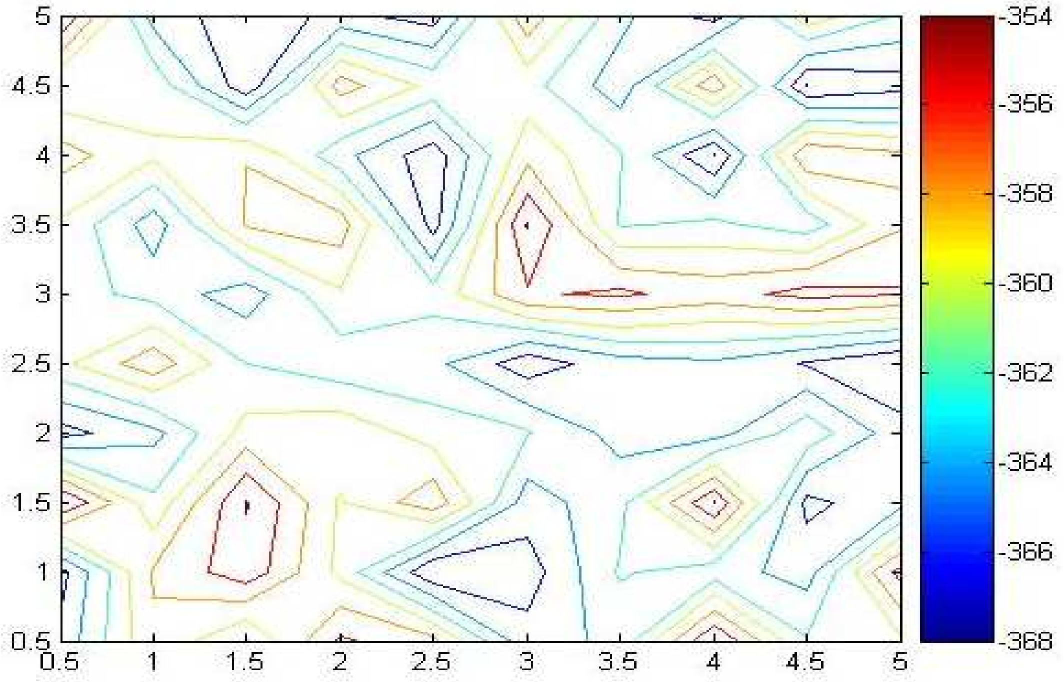


Fig. 14: VO_2 energy contour plot

Additional trials with alternate settings were also done to better understand the behavior of the PSO algorithms in this application. As expected, an increase in the number of particles and groups yields longer computation times. The RG-PSO algorithm with 30 groups of 3 particles (90 total particles) resulted in a convergence time of 48485 seconds for the VO_2 structure, and 10393 seconds for FeTiH. These computation times are significantly longer than those required by BFGS. Each of the 30 groups settled on a different local optimum location, but the algorithm again failed to find the same arrangement that was found with BFGS. Trials with 150 particles and 300 particles with a group size of five were also attempted. For the FeTiH structure, results were similar to the 90 particle case, with longer computation times and convergence to many local optima. Relaxation for the VO_2 material fails to converge if a very large number of particles is used; with computation times reaching in excess of 30 hours, none of the groups settle on a global optimum. The standard PSO algorithm performed slightly better than RG-PSO when using 100 and 300 particles. Computation times did not increase as drastically, but the optimum location that was found did not match the BFGS result.

Table 17 shows that in some cases a more optimal energy level was found using the PSO algorithms. Again, this is due to the global searching nature of PSO, as opposed to the local search of BFGS. In all cases, the optimum locations that were found had energy levels very close to those found by BFGS. As mentioned previously, AG-PSO was found to be less effective than the RG-PSO algorithm, so it was omitted from many of the testing trials in this section. It should be noted that although data is presented for relaxation from initial state and final state, the starting position that is used has no effect on the PSO algorithms. For each material, the initial and final states are located on the same potential energy surface, but the different starting locations lead to different results with the BFGS algorithm. The PSO methods, however, search the entire space using a random initial distribution of particles, regardless of whether the relaxation is for initial state or final state. The different results seen in Table 17 are simply due to the random element of the PSO algorithms.

Table 15: Relaxation computation time (s), for initial and final states

		BFGS	PSO	RG-PSO	AG-PSO
FeTi	Initial	983	1204	802	829
	Final	568	789	479	459
VO_2	Initial	8940	10918	4857	4803
	Final	35837	38060	5221	5640
FePt	Initial	21	102	21	21
	Final	72	241	72	71

Table 16: Atom locations and minimum energies for FeTi, VO_2, and FePt

		BFGS	PSO	Final Energy	
				BFGS	PSO
FeTi Initial	Fe	(0.0,0.0,0.0)	(3.06,0.0172,-0.549)	-346.52938	-346.5294
	Fe	(0.0,4.29,0.0)	(5.38,0.651,3.14)		
	Ti	(2.79,2.14,4.14)	(2.81,4.24,5.28)		
	Ti	(2.79,6.44,4.14)	(4.23,-1.10,0.686)		
	H	(0.0,0.0,4.14)	(0.117,2.94,4.03)		
	H	(0.0,4.29,4.14)	(4.07,1.59,2.37)		

FeTi Final	Fe	(0.0,-0.408,0.0)	(2.35,3.29,2.37)	-346.7658	-346.7305
	Fe	(0.0,4.7,4.14)	(5.55,1.68,4.02)		
	Ti	(2.79,-0.104,4.14)	(2.43,1.05,2.54)		
	Ti	(2.79,4.39,0.0)	(2.43,0.88,1.50)		
	H	(0.0,2.14,2.07)	(0.885,1.26,0.402)		
	H	(0.0,2.14,6.22)	(3.33,1.94,1.11)		
VO_2 Initial	V	(0.0,0.0,0.0)	(4.101,4.35,1.31)	-367.94	-366.019
	V	(4.52,4.52,2.83)	(4.80,1.24,4.41)		
	V	(0.0,0.0,5.66)	(2.18,1.57,4.66)		
	V	(4.52,4.52,8.49)	(3.15,0.364,4.55)		
	O	(1.73,1.73,2.83)	(2.09,1.87,0.568)		
	O	(7.31,7.31,2.83)	(5.21,1.83,0.935)		
	O	(6.25,2.79,0.0)	(4.24,3.68,1.69)		
	O	(2.79,6.25,0.0)	(0.56,2.09,1.29)		
	O	(1.73,1.73,8.49)	(2.78,3.99,1.90)		
	O	(7.31,7.31,8.49)	(3.36,0.119,3.93)		
	O	(6.25,2.79,5.66)	(4.05,1.77,1.44)		
	O	(2.79,6.25,5.66)	(1.82,-0.373,4.39)		
VO_2 Final	V	(-0.002,0.98,-0.002)	No Convergence	-368.067	-353.77
	V	(4.3,5.28,2.69)	No Convergence		
	V	(0.002,0.98,5.39)	No Convergence		
	V	(4.3,5.28,8.08)	No Convergence		

	O	(1.73,2.7,2.7)	No Convergence		
	O	(6.88,7.87,2.69)	No Convergence		
	O	(6.03,3.56,-0.001)	No Convergence		
	O	(2.58,7.005,-0.0004)	No Convergence		
	O	(6.89,-0.75,8.08)	No Convergence		
	O	(1.72,11.32,8.08)	No Convergence		
	O	(6.02,3.56,5.39)	No Convergence		
	O	(2.57,7.007,5.39)	No Convergence		
FePt Initial	Fe	(2.57,0.0,3.51)	(1.48,1.32,1.89)	-281.445	-281.505
	Fe	(2.57,2.57,0.0)	(1.01,2.57,0.898)		
	Pt	(0.0,0.0,0.0)	(1.89,2.36,-2.50)		
	Pt	(0.0,2.57,3.51)	(3.33,4.24,4.38)		
FePt Final	Fe	(2.57,0.0,3.51)	(0.468,1.83,2.18)	-280.8954	-280.8955
	Fe	(0.0,2.57,3.51)	(3.11,3.64,1.45)		
	Pt	(0.0,0.0,0.0)	(2.74,1.25,2.75)		
	Pt	(2.57,2.57,0.0)	(2.91,0.737,2.69)		

Table 17: Total energy at optimal location found by each algorithm

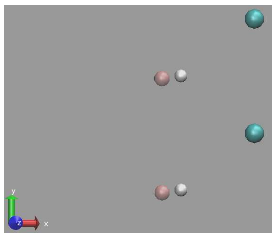
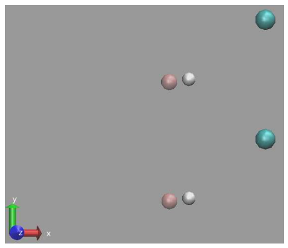
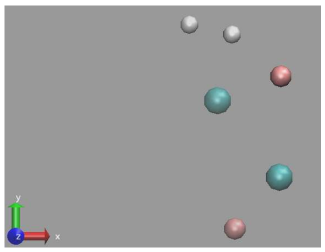
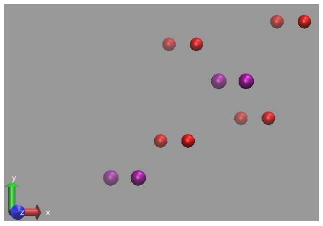
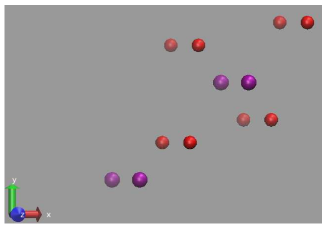
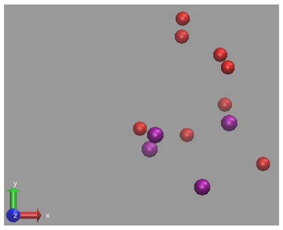
	BFGS	PSO	RG-PSO
FeTi Initial	-346.52938	-346.5294	-346.513
FeTi Final	-346.7658	-346.7305	-346.729
VO_2 Initial	-367.94	-366.019	-366.019

VO_2 Final	-368.067	-353.77	-363.718
FePt Initial	-281.445	-281.505	-281.445
FePt Final	-280.8954	-280.8955	-280.895

A graphical representation of the relaxed positions found using BFGS and RG-PSO is shown in table

Table 18. The FePt structure is not shown because it is relatively trivial compared to the FeTiH and VO_2 examples, and AG-PSO results are omitted because they are similar or worse than those obtained with RG-PSO. The atom distribution in the relaxed states confirms the findings of the previously shown data. BFGS relaxation converges to a local optimum with atom positions very similar to the starting positions; the difference between the BFGS relaxed state and the initial distribution is not noticeable, as each atom experiences very slight movement. After relaxation with the RG-PSO, however, we can see a drastically different atom distribution. These positions represent the global optimum energy level, or at least a different local optimum than that found by BFGS. Since the starting position has no effect on the RG-PSO algorithm, it is free to search the space for any locations of minimum energy, resulting in an amorphous material. Additional coordinates and figures of optima found by other groups of the RG-PSO algorithm are shown in appendix sections 6.9 and 6.10.

Table 18: Relaxed positions with BFGS and RG-PSO

	Starting Positions	After relaxation with BFGS	After relaxation with RG-PSO
FeTiH			
VO_2			

The RG-PSO result seen in

Table 18 is not useful for our transition simulation application. Although a lower energy state may be found, the atom positions no longer represent the initial state that is required for simulation. In order to find the same local optimum that is found with the BFGS algorithm, we can initially force all the PSO particles into the starting structure position instead of using a random distribution. This technique greatly increases the likelihood that the same local optimum and relaxed position will be found. Using the standard PSO algorithm with ten particles, convergence was achieved in 1840 seconds for the initial FeTiH structure, and 664 seconds for the final FeTiH structure. Although these computation times are higher than the 983 seconds and 568 seconds required by BFGS for the respective cases, the algorithm converged to the expected local optimum, and the same relaxed structure was found. RG-PSO with 15 particles and five groups was less consistent, requiring only 764 seconds for relaxation of the initial structure and 869 seconds for the final structure, but again the desired local optimum was found. Some typical results of the RG-PSO search with forced initial positions for the FeTiH and VO_2 structures can be seen in section 6.11, where the positions and energy values in the first row show the same result that is found by BFGS, and the following rows show additional optima that were found by other groups.

Although the methods worked as intended and in some cases found a global optimum or a better local optimum than BFGS, the resulting structures represent an amorphous material that is different from the crystal structure found with BFGS relaxation. It is possible to find the same result as BFGS by forcing the particles into a specific starting position, but the computation times are generally longer than BFGS with the benefit that more information about the search space is obtained through the additional optima that are found.

Table 19: Pseudo-code for PSO algorithm implemented with QE

```
**INPUT (from file)**: array containing the initial atom locations $\{\vec{x}_0\}$, energy of the system $\{f(\vec{x}_i)\}$,
array containing components of the gradient $\{\vec{g}_i\}$, number of particles $\{N\}$
**INPUT (within module)**: convergence threshold $\{energy\_thr\}$, upper and lower search space
boundaries $\{b_u, b_l\}$, constants for local and global directions $\{c_1, c_2\}$, movement restriction $\{J\}$,
maximum number of iterations $\{Maxi\}$

**OUTPUT**: set of $i$ optimized locations $\{\vec{x}_i\}$
FOR $i = 1 : Maxi$
FOR $i = 1 : N$
    $X_i = rand() * (b_u - b_l) + b_l$
    $pbest_i = X_i$
    $V_i = 0$
    $f_i = f(\vec{x}_i)$
END
$gbest = \min(f_i)$
FOR $j = 1 : N$
    $newV_j = \vartheta * V_j + c_1 * rand() * (pbest_j - x_j) + c_2 * rand() * (gbest - x_j)$
    $newX_j = X_j + newV_j$
    CALL $electrons(newX)$
    $newf_j = etot$
END
$X = newX$
$V = newV$
$gbest = \min(f_i)$
IF $\min(f_i) > \min(newf_i)$
    $gbest = \min(newf_i)$
END
FOR $n = 1 : N$
    IF $newf_n < f_n$
        $pbest_n = X_n$
    END
END
FOR $i = 1 : N$
    $\vec{x} = \vec{x}_i$
    $f_i = f(\vec{x}_i)$
END
$f_j = newf_j$
END
```

5. CONCLUSIONS AND RECOMMENDATIONS

In this thesis, the general shortcomings of the classic particle swarm optimization method were demonstrated, in particular with regard to multimodal functions. Some of these drawbacks were addressed using the novel Random Group and Active Group particle swarm optimization methods by splitting the initial population of particles into smaller groups and using two different techniques for position and velocity updates of particles. Through the use of a variety of test functions, it was shown that the RG-PSO algorithm yields an improvement not only in accuracy, but also in computation time for some functions, while the AG-PSO method also provides better accuracy than the standard PSO algorithm. The potential of these variations of the PSO method for use within the Quantum Espresso software was also demonstrated. Compared to the native BFGS method, a more favorable energy state and faster convergence was achieved in some cases.

Further, a geometry-guided phase transition pathway search method for finding intermediate states in the phase transition of crystals was proposed. A periodic surface model is used to build crystals parametrically for ease of construction and modification. The transition pathway is then estimated by interpolation of periodic planes, and atom positions are determined by intersection of the loci surfaces. The estimation can provide a good initial guess of MEP for physics-based transition path and saddle point search methods such that the risk of being trapped in a local minimum energy path is reduced. To enable this integrated computer-aided transition pathway design, methods were developed for finding correspondence of atom locations and periodic planes in the initial and final states of a crystalline material's unit cell. A heuristic global optimization approach is taken to reduce the complexity of searching correspondence. Once the surface models in the initial and final states are matched, the developed surface linear interpolation and potential-driven surface interpolation methods are used to make predictions about how each atom will move.

The proposed approaches are intended to integrate geometry and physics information in materials modeling and simulation. Further exploration of the intrinsic relation between the two that goes beyond the simple observations is meaningful. Observations from this thesis include that geometric structures and physical properties in nanoscale materials have connections and even one-to-one mappings. Metamorphosis in geometry can integrate more physics of phase transitions. Structures with strongly bonded atoms can simplify the computation of geometric morphing. Modeling the interactions among geometric entities can help simulate physical phenomena more efficiently.

Numerical error is a challenge in our proposed approach. When the angles between intersecting planes are small and rotation is involved during the surface interpolation, the numerical error to compute intersections may become significant. The discretization of the 3D space to generate fine-grained models in our implicit modeling scheme is then essential to keep errors small, which will increase the computation time. One possible way to alleviate this is to define planes with intersecting angles as large as possible, such as the y-z, x-z, and x-y planes. In this thesis we have shown that for some cases the saddle point is identified more accurately using our techniques.

5.1 Conclusions

A few major conclusions can be drawn from the work presented in this thesis.

- The simmulated annealing global optimization algorithm is effective for searching plane correspondence. In a general case, as well as the examples shown, too many atoms may be present to check all possible correspondence combinations. Considering that three planes are required to represent each atom, the number of possibilities increases quickly due to combinatorial complexity. Depending on the values that are used for T and ΔT, the algorithm can converge quickly and settle on a combination that is fairly close to optimal. With this application in particular, the global optimum is not absolutely required. Even a local optimum is able to generate an initial guess of atom movement that is more favorable than the linear prediction used with the standard NEB method. Therefore, optimization using the SA algorithm is faster than an exhaustive search and still generates acceptable results.
- Surface linear interpolation is generally more effective for atom movement prediction than potential-driven surface interpolation, but both methods can be improvements over the standard NEB method with a linear initial guess. It was found that the potential-driven method tends to predict individual movement of atoms, rather than the simultaneous movement we see with surface linear interpolation. This causes a less accurate guess of initial path and as a result a higher error in the prediction of activation energy. Although the surface linear interpolation method may predict some atom positions that are not physically feasible, it yielded more accurate results for the examples that were tested. In some cases, however, both methods are preferred over the native NEB method, which may not converge using its initial guess of a linear path.
- Both RG-PSO and AG-PSO show better accuracy than the standard PSO method.

Particularly for some of the multimodal test functions that were used, the PSO method shows low success rates and was likely to settle at a local optimum. Although computation times are higher in some cases, RG-PSO and AG-PSO have success rates of 100 percent for most of the test function trials. Additionally, using the RG-PSO and AG-PSO methods, we can obtain more information about other optima in the search space, rather than just the singular global optimum value we get with standard PSO. In general, using the newly proposed methods in this thesis, we can get higher accuracy and more information in exchange for slightly longer computation times.

- RG-PSO is more effective and converges faster than AG-PSO. It was found that for all of the test functions, as well as with the Quantum Espresso implementation, RG-PSO was much more useful. With the AG-PSO algorithm, the movement away from neighboring groups inhibits convergence, as it often results in the groups moving away from an optimum position. This effect can be counteracted by using a very low c_3 value; however, as c_3 approaches zero, the AG-PSO algorithm becomes identical to RG-PSO. Although the accuracy with AG-PSO was also high, it does not show any advantages over RG-PSO, and therefore it was omitted from some of the test runs.
- Global optimization methods are not ideal for implementation with Quantum Espresso for the relaxation portion of pathway search. Using the default BFGS method within Quantum Espresso, we generally find a local optimum which is located very close to the starting position. This allows us to find a relaxed state that has atom positions very similar to the starting and ending locations obtained from literature. The PSO algorithms, on the other hand, are a global search across the entire search space that is defined within the algorithm. By default, the particles are initially

distributed randomly throughout this search space, so the start and end locations are not used. There is a chance that a local optimum is found that matches the result from the BFGS algorithm, but in most cases the global optimum or a different local optimum is found and the resulting atom locations represent an amorphous material rather than a crystal structure. If the starting positions of some or all of the particles are specified corresponding to the locations we find in literature, the algorithm can return the same result as the BFGS method. Using RG-PSO, we can obtain this particular local optimum, as well as the global optimum or other local optima. By implementing this modification it is possible to use the PSO methods to find the same local optimum as the BFGS method, but the technique is slower than the standard BFGS algorithm and the advantage of global search is not utilized.

5.2 Recommendations

In future research, we would like to extend our techniques to more applications and test how effective our methods are, with opportunities to further refine our approach. The correspondence search and surface interpolation methods are designed to work with any crystal structure, and have been demonstrated with three examples. However, it may be beneficial to test these methods using additional structures to show their robustness. It may also be possible to extend the interpolation to other, non-crystal materials that can be modeled using periodic surfaces.

It will also be useful to make the proposed PSO variations more robust and useful for general cases. Currently a useful result is only obtained if the starting distribution of particles is adjusted manually, or if the search space is constrained such that the number of optima is reduced. Both of these techniques require relatively detailed knowledge about the particular example that is being studied. To create a more useful tool, the amount of user input that is required should be similar to the BFGS method, so some refinement of the PSO methods is required to find the expected local optimum more reliably.

6. APPENDIX

6.1 STANDARD PSO ALGORITHM

```
clear all;

N=100;          %Number of particles.

xbl=-10;        %Upper and lower search space boundaries in x and y.
xbu=10;
ybl=-10;
ybu=10;
Maxi=500;       %Maximum number of iterations.
cl=1.5;         %Constants for local and global terms.
cg=2;
theta=0.5;      %Movement restriction.


for g=1:N
    x(1,g)=rand()*(xbu-xbl)+xbl;
    x(2,g)=rand()*(ybu-ybl)+ybl;
end

for i=1:N                  %Assign random initial positions and initial zero velocities.
    v(1:2,i)=0;
    f(1,i)=(x(1,i)+2*x(2,i)-7)^2+(2*x(1,i)+x(2,i)-5)^2;          %Objective function.
end

y=randsample(length(x),length(x));
y=y';

for i=1:N
    newx(:,i)=x(:,y(i));                %Re-sorts the positions vector randomly.
    newf(1,i)=f(1,y(i));
    pbest(:,i)=newx(:,i);
end
x=newx;
f=newf;

[C,I]=min(f);
gbest=x(:,I);

for i=1:Maxi
```

```
        for j=1:N
            newv(:,j)=theta*v(:,j)+cl*rand()*(pbest(:,j)-x(:,j))+cg*rand()*(gbest-x(:,j));
            newx(:,j)=x(:,j)+newv(:,j);
            newf(1,j)=(x(1,j)+2*x(2,j)-7)^2+(2*x(1,j)+x(2,j)-5)^2;
        end
        x=newx;
        v=newv;
        [C,I]=min(f);
        gbest=x(:,I);
        if C<min(newf)
            [C,I]=min(newf);
            gbest=x(:,I);
        end
        for n=1:N
            if newf(n)<f(n)
                pbest(:,n)=x(:,n);
            end
        end
        f=newf;
        if min(f)<0.01          %Ends the loop if objective function for any particles is
            break               %close to 0 (global min).
        end
    end


    [C,I]=min(f);
    I
    x(:,I)
    min(f)

    %Create grid to show function and particles.

    [b,d]=meshgrid(-5.5:.1:5.5,-5.5:.1:5.5);

    n=2;
    z=(b+2*d-7).^2+(2*b+d-5).^2;
    surfc(b,d,z,'EdgeColor','none');

    hold on;
    for i=1:N
        n=2;
        z=(x(1,i)+2*x(2,i)-7)^2+(2*x(1,i)+x(2,i)-5)^2;
        plot3(x(1,i),x(2,i),z,'r.','MarkerSize',20)
        shading interp
    end
    hold off;
```

6.2 RG-PSO ALGORITHM

```
%Modified hybrid PSO algorithm for implementation with crystal phase transition.
%The initial population of particles is divided into sub-groups, and particles in
%each sub-group update their positions based on personal best and sub-group
%best. Additionally, the sub-group best is updated with criticality to prevent sub-
%groups from getting too close to each other and finding the same optimum.


clear all;

N=100;              %Number of particles.
gsize=5;            %Number of particles in each sub-group.

xbl=-10;            %Upper and lower search space boundaries in x and y.
xbu=10;
ybl=-10;
ybu=10;

Maxi=1000;          %Maximum number of iterations.
cl=1.7;             %Constants for local and global terms.
cg=1.7;
theta=0.6;          %Movement restriction (may also lower this with iterations).
td=0.01*xbu;        %Threshold distance (distance between avg. position of
                    %particle groups that is "too close").
maxcrit=4;          %Maximum allowed criticality value before relocation.

for i=1:N                            %Assign random initial positions and initial
    x(1,i)=rand()*(xbu-xbl)+xbl;     %zero velocities. Evaluate objective function at
    x(2,i)=rand()*(ybu-ybl)+ybl;     %each position.
    v(1:2,i)=0;
    f(1,i)=(x(1,i)+2*x(2,i)-7)^2+(2*x(1,i)+x(2,i)-5)^2;
end


if floor(N/gsize) == N/gsize         %Make sure number of groups is rounded up
    groups = N/gsize;                %to nearest integer.
else
    groups = floor(N/gsize)+1;
end

y=randsample(length(x),length(x));
y=y';                                      %Transpose
```

```
for i=1:N
     newx(:,i)=x(:,y(i));       %Re-sorts the positions vector randomly.
     newf(1,i)=f(1,y(i));
     pbest(:,i)=newx(:,i);
end
x=newx;
f=newf;

for i=1:groups                       %Finds the best particle in each group and stores it
                                     %in "gbest" array.
     [C,I]=min(f(((i*gsize)-gsize+1):i*gsize));     %Objective function for gbest
                                                    %locations is stored in "fgbest".

     gbest(:,i)=x(:,I);
     fgbest(i)=min(f(((i*gsize)-gsize+1):i*gsize));
end


crit=zeros(1,groups);                   %Sets up an initial zero matrix for "criticality".
for i=1:Maxi      %Update positions and velocities, re-evaluate objective function.
     for j=1:groups                     %Each group uses its own gbest for updates.
          for n=(j*gsize-gsize+1):(j*gsize) %"j*gsize-gsize+1:j*gsize" forces search
                                            %by %group, i.e. 1-5, 6-10, 11-15 etc.
        v(:,n)=theta*v(:,n)+cl*rand()*(pbest(:,n)-x(:,n))+cg*rand()*(gbest(:,j)-x(:,n));
        x(:,n)=x(:,n)+v(:,n);
          end
          for m=1:2
               avglocation(m,j)=(sum(x(m,(j*gsize-gsize+1):(j*gsize))))/gsize;
          end
     end
     for j=1:groups
          for k=j+1:groups
             dist(j)=sqrt(sum((avglocation(:,k)-avglocation(:,j)))^2); %Find distance
                            between average locations of each group.
               if dist(j)<td   %If distance is less than threshold, increase criticality of
                    crit(k)=crit(k)+1;   %that group by one. If criticality is large
                                         %enough, relocate all particles in that
                    if crit(k)>=maxcrit            %group randomly.
                         for h=(k*gsize-gsize+1):(k*gsize)
                              x(1,h)=rand()*(xbu-xbl)+xbl;
                              x(2,h)=rand()*(ybu-ybl)+ybl;
                              v(1:2,h)=0;
                         end
                    end
               end
          end
```

```
    end
    for n=1:N
        newf(n)=(x(1,n)+2*x(2,n)-7)^2+(2*x(1,n)+x(2,n)-5)^2;
    end
    for a=1:groups
        for k=((a*gsize)-gsize+1):(a*gsize)
            if newf(k)<fgbest(a)       %Replace gbest/pbest values if better value
                gbest(:,a)=x(:,k);     %is found.
                fgbest(a)=newf(k);
            end
        end
    end
    for n=1:N
        if newf(n)<f(n)
            pbest(:,n)=x(:,n);
        end
    end
    f=newf;
    if min(f)<0.01   %Ends the loop if objective function for any particles is close
        i            %to 0 (global min).
        break
    end
    i
end

for j=1:groups
    [C,I]=min(f((j*gsize-gsize+1):(j*gsize)));        %Displays x and y coordinates of
   location(1:2,j)=x(1:2,I+j*gsize-gsize);            % best particle in each group, as
    location(3,j)=min(f((j*gsize-gsize+1):(j*gsize)));   %well as objective function
                                                         %at those locations.
end
min(location(3,:))
[C,I]=min(location(3,1:groups));

%Commands to create grid and show function and particle locations marked in
red.
[b,d]=meshgrid(-10:.1:10,-10:.1:10);
z=(b+2*d-7).^2+(2*b+d-5).^2;
surfc(b,d,z,'EdgeColor','none');

hold on;
for i=1:groups
    z=(location(1,i)+2*location(2,i)-7)^2+(2*location(1,i)+location(2,i)-5)^2;
    plot3(location(1,i),location(2,i),z,'r.','MarkerSize',20)
    shading interp
end
```

```
hold off;
```

6.3 AG-PSO ALGORITHM

```
%Modified hybrid PSO algorithm for implementation with crystal phase transition.
The initial %population of particles is divided into sub-groups, and particles in
each sub-group update their %positions based on personal best and sub-group
best. Additionally, the sub-group best is %influenced by the (three) closest other
sub-groups and moves away from them. To do this,
%additional, negative terms are implemented in the velocity/position updates that
cause the group %to move away rather than towards a particular location.

clear all;

N=100;            %Number of particles.
gsize=5;          %Number of particles in each sub-group.

xbl=-10;          %Upper and lower search space boundaries in x and y.
xbu=10;
ybl=-10;
ybu=10;

Maxi=1000;        %Maximum number of iterations.
cl=1.7;           %Constants for local and global terms.
cg=1.7;
cn=0.1;           %Constant for terms from neighbors.
theta=0.6;        %Movement restriction (may also lower this with iterations).
neighbors=3;           %Number of neighboring groups to be considered.


for i=1:N                               %Assign random initial positions and initial
    x(1,i)=rand()*(xbu-xbl)+xbl;        %zero velocities. Evaluate objective function at
    x(2,i)=rand()*(ybu-ybl)+ybl;        %each position.
    v(1:2,i)=0;
    f(1,i)=(x(1,i)+2*x(2,i)-7)^2+(2*x(1,i)+x(2,i)-5)^2;
end

if floor(N/gsize) == N/gsize
    groups = N/gsize;
else
    groups = floor(N/gsize)+1;
end

y=randsample(length(x),length(x));
y=y';                                   %Transpose
```

```
for i=1:N
    newx(:,i)=x(:,y(i));                    %Re-sorts the positions vector randomly.
    newf(1,i)=f(1,y(i));
    pbest(:,i)=newx(:,i);
end
x=newx;
f=newf;

for i=1:groups
    [C,I]=min(f(((i*gsize)-gsize+1):i*gsize));
    gbest(:,i)=x(:,I);
    fgbest(i)=min(f(((i*gsize)-gsize+1):i*gsize));
end


theta=0.6;
for i=1:Maxi
    if i==0.2*Maxi | i==0.4*Maxi | i==0.6*Maxi | i==0.8*Maxi   %This "if" condition
        theta=theta-0.1                         % gradually reduces the value of theta.
    end
    for j=1:groups
        for m=1:2
            avglocation(m,j)=(sum(x(m,(j*gsize-gsize+1):(j*gsize))))/gsize;
        end
    end
    for j=1:groups
        for k=1:groups
            dist(j,k)=sqrt(sum((avglocation(:,k)-avglocation(:,j)))^2);
        end
    end
    for j=1:2:(groups*2)
        M(j,:)=dist(ceil(j/2),:);          %M is matrix of distances between groups
        M(j+1,:)=1:groups;                 %and indices.
        B(:,j:(j+1))=sortrows(M(j:(j+1),:)');     %B is matrix of sorted distances,
    end                                  %from min to max, and corresponding indices.

    for j=1:groups
        for n=(j*gsize-gsize+1):(j*gsize)
            vadd(1:2,n)=zeros;           %"vadd" is the sum of additional terms
                                         %from neighboring locations.
            for k=2:(neighbors+1)       %It is subtracted from the standard "v" term
                                        %in each update.
                vadd(:,n)=vadd(:,n)+(cn*rand()*(avglocation(:,B(k,j*2))-x(:,n)));
            end
```

```
                v(:,n)=theta*v(:,n)+cl*rand()*(pbest(:,n)-x(:,n))+cg*rand()*(gbest(:,j)-
                x(:,n))-vadd(:,n);
                x(:,n)=x(:,n)+v(:,n);
            end
        end

        for n=1:N
            newf(n)=(x(1,n)+2*x(2,n)-7)^2+(2*x(1,n)+x(2,n)-5)^2;
        end
        for a=1:groups
           for k=((a*gsize)-gsize+1):(a*gsize)
                if newf(k)<fgbest(a)            %Replace gbest/pbest values if better
                    gbest(:,a)=x(:,k);          %value is found.
                    fgbest(a)=newf(k);
                end
            end
        end
        for n=1:N
            if newf(n)<f(n)
                pbest(:,n)=x(:,n);
            end
        end
        f=newf;
        if min(f)<0.01                       %Ends the loop if objective function for
            i                                %any particles is close to 0 (global min).
            break
        end
        i
    end


    for j=1:groups
       [C,I]=min(f((j*gsize-gsize+1):(j*gsize)));          %Displays x and y coordinates
                                                            %of best particle in each group,
        location(1:2,j)=x(1:2,I+j*gsize-gsize);             %as well as objective function
                                                            %at those locations.
        location(3,j)=min(f((j*gsize-gsize+1):(j*gsize)));
    end
    location
    min(location(3,:))
    [C,I]=min(location(3,1:groups));

    %Commands to create grid and show function and particle locations marked in
    %red.
    [b,d]=meshgrid(-10:.1:10,-10:.1:10);
    z=(b+2*d-7).^2+(2*b+d-5).^2;
```

```
surfc(b,d,z,'EdgeColor','none');

hold on;
for i=1:groups
     z=(location(1,i)+2*location(2,i)-7)^2+(2*location(1,i)+location(2,i)-5)^2;
     plot3(location(1,i),location(2,i),z,'r.','MarkerSize',20)
     shading interp
end
hold off;
```

6.4 *PSO MODULE (FORTRAN)*

```
!PSO algorithm used to optimize the atom positions. Uses
some initialization and subroutines from the bfgs module,
!but the optimization algorithm is replaced with PSO.

MODULE bfgs_module

USE kinds,       ONLY : DP
USE io_files,    ONLY : iunbfgs, prefix
USE constants,   ONLY : eps16
USE basic_algebra_routines

IMPLICIT NONE
PRIVATE
PUBLIC :: bfgs, terminate_bfgs
PUBLIC :: bfgs_ndim, epsilon, N, NN, MX, ITRN, NPRN, &
          trust_radius_max, trust_radius_min,
          trust_radius_ini, w_1, w_2

SAVE

CHARACTER (len=8) :: fname="energy"

REAL(DP), ALLOCATABLE :: &
    pos(:),             &
    grad(:),            &
    pos_p(:),           &
    grad_p(:),          &
    inv_hess(:,:),      &
    metric(:,:),        &
    h_block(:,:),       &
    hinv_block(:,:),    &
    step(:),            &
    step_old(:),        &
    pos_old(:,:),       &
    grad_old(:,:),        &
    pos_best(:),          &
    X(:,:),               &
    V(:,:),               &
    F(:),                 &
    A(:),                 &
    BST(:),               &
    V1(:),                &
    V2(:),                &
    V3(:),                &
    V4(:),                &
    XX(:,:),              &
    VI(:)
REAL(DP) :: &
    trust_radius,         &
    trust_radius_old,     &
    energy_p
INTEGER :: &
    scf_iter,             &
    bfgs_iter,            &
    gdiis_iter
LOGICAL :: tr_min_hit, conv_bfgs

INTEGER ::  bfgs_ndim, IU, NSTEPpso, IV, N, NN, MX, ITRN,
NPRN

REAL(DP)  :: epsilon, trust_radius_max, trust_radius_min,
trust_radius_ini, w_1, w_2

CONTAINS

SUBROUTINE bfgs(  pos_in, h, energy, grad_in, fcell,  &
fixion, scratch, stdout, energy_thr, grad_thr, cell_thr, &
energy_error, grad_error, cell_error, istep, nstep, &
step_accepted, stop_bfgs, lmovecell)

COMMON /RNDM/IU
COMMON /KFF/KF,NFCALL,FTIT
CHARACTER *70 FTIT
REAL(DP) :: FI, BEST, II, NF, NFBEST, RANDS, IV, B, PI

REAL(DP),            INTENT(INOUT) :: pos_in(:)
REAL(DP),            INTENT(OUT)   :: h(3,3)
REAL(DP),            INTENT(INOUT) :: energy
REAL(DP),            INTENT(INOUT) :: grad_in(:)
REAL(DP),            INTENT(INOUT) :: fcell(3,3)
INTEGER,             INTENT(IN)    :: fixion(:)
CHARACTER(LEN=*),    INTENT(IN)    :: scratch
INTEGER,             INTENT(IN)    :: stdout
```

```
REAL(DP),           INTENT(IN)    :: energy_thr, grad_thr, &
                                     cell_thr
INTEGER,            INTENT(OUT)   :: istep
INTEGER,            INTENT(IN)    :: nstep
REAL(DP),           INTENT(OUT)   :: energy_error, &
                                     grad_error, cell_error
LOGICAL,            INTENT(OUT)   :: step_accepted, &
                                     stop_bfgs
LOGICAL,            INTENT(IN)    :: lmovecell

INTEGER  :: ni,iii,ji,k,nat, M, KF, LCOUNT, NFCALL, IU, &
ITER, IN, JJ, J, I
LOGICAL  :: lwolfe
REAL(DP) :: dEOs, den
REAL(DP) :: hinv(3,3),g(3,3),givn(3,3),garbage, ginv(3,3)
REAL(DP) :: A1, A2, A3, W, SIGMA, FFMIN, RAND
REAL :: FMIN

FMIN = 1.0E30
A1 = 1.7D00
A2 = 1.7D00
A3 = 0.005D00
W  = 0.6D00
SIGMA = 1.D-03
epsilon             = 1.D-08   !Accuracy needed for termination
N                   = 1000
NN                  = 1000
MX                  = 100
NSTEPpso            = 15
ITRN                = 10000
NPRN                = 500

ni=SIZE( pos_in ) + 9
nat=size (pos_in) / 3

ALLOCATE( V(N,MX))
ALLOCATE( F(N))
ALLOCATE( A(MX))
ALLOCATE( BST(MX))
ALLOCATE( V1(MX))
ALLOCATE( V2(MX))
ALLOCATE( V3(MX))
ALLOCATE( V4(MX))
ALLOCATE( XX(N,MX))
ALLOCATE( VI(MX))
ALLOCATE( X(N,MX))
ALLOCATE( pos(   ni) )
ALLOCATE( grad(  ni) )
ALLOCATE( grad_old( ni, bfgs_ndim) )
ALLOCATE( pos_old( ni, bfgs_ndim) )
ALLOCATE( inv_hess( ni, ni ) )
ALLOCATE( pos_p(   ni) )
ALLOCATE( grad_p(  ni) )
ALLOCATE( step(    ni) )
ALLOCATE( step_old(ni) )
ALLOCATE( pos_best(ni) )
ALLOCATE( hinv_block( ni-9, ni-9) )
ALLOCATE( metric( ni, ni ) )
CALL invmat(3, h, hinv, garbage)
hinv_block = 0.d0
FORALL ( k=0:nat-1, iii=1:3, ji=1:3 )
hinv_block(iii+3*k,ji+3*k) = hinv(iii,ji)
g=MATMUL(TRANSPOSE(h),h)
CALL invmat(3,g,ginv,garbage)
metric = 0.d0
FORALL ( k=0:nat-1,   iii=1:3, ji=1:3 ) metric(iii+3*k,
ji+3*k) = g(iii,ji)
FORALL ( k=nat:nat+2, iii=1:3, ji=1:3 ) metric(iii+3*k,
ji+3*k) = 10.0* ginv(iii,ji)
pos = 0.0
pos(1:ni-9) = pos_in
IF (lmovecell) FORALL( iii=1:3, ji=1:3) pos( ni-9 +
ji+3*(iii-1) ) = h(iii,ji)
grad=0.0
grad(1:ni-9)=grad_in
IF (lmovecell) FORALL( iii=1:3, ji=1:3) grad( ni-9 +
ji+3*(iii-1) ) = fcell(iii,ji)

IF ( lmovecell ) fname="enthalpy"
CALL read_bfgs_file( pos, grad, fixion, energy, scratch, &
     ni, stdout )
scf_iter = scf_iter+1
istep    = scf_iter

energy_error = ABS( energy_p - energy )
grad_error = MAXVAL( ABS( MATMUL( TRANSPOSE(hinv_block),
     grad(1:ni-9)) ) )
conv_bfgs = energy_error < energy_thr
conv_bfgs = conv_bfgs .AND. ( grad_error < grad_thr )

IF( lmovecell) THEN
   cell_error = MAXVAL( ABS( grad(ni-8:ni) ) )
   conv_bfgs = conv_bfgs .AND. ( cell_error < cell_thr )
END IF
stop_bfgs = conv_bfgs .OR. ( scf_iter >= nstep)
IF (stop_bfgs) GOTO 1000
```

```
WRITE( UNIT = stdout, FMT = '(/,5X,"number of scf &
cycles",T30,"= ",I3)' ) scf_iter
WRITE( UNIT = stdout,  FMT = '(5X,"number of bfgs &
steps",T30,"= ",I3,/)' ) bfgs_iter
IF ( scf_iter > 1 ) WRITE( UNIT = stdout, FMT = '(5X,A," &
old",T30,"= ",F18.10," Ry")' ) fname, energy_p
WRITE( UNIT = stdout, FMT = '(5X,A," new",T30,"= ",F18.10,"&
Ry",/)' ) fname,energy

IF ( ( energy > energy_p ) .AND. ( scf_iter > 1 ) ) THEN
        step_accepted = .FALSE.
WRITE( UNIT = stdout, &
& FMT = '(5X,"CASE: ",A,"_new > ",A,"_old",/)' ) fname,fname
ELSE
!------------------------------------------------------------
!PSO algorithm starts here.

CALL FSELECT(KF,M,FTIT)

FFMIN=1.D30
LCOUNT=0
NFCALL=0
IU=5121

DO I=1,N
   DO J=1,M
   CALL RANDOM(RAND)
        X(I,J)=(RAND)*5    !Generates RANDOM(0,5)
   ENDDO
   F(I)=1.0D30
ENDDO

DO I=1,N
   DO J=1,M
       CALL RANDOM(RAND)
       V(I,J)=(RAND-0.5D+00)
   ENDDO
ENDDO

WRITE(*,*)'CHECKING VALUES OF N, NPRN', N, NPRN
WRITE(*,*)'FINISHED INITIAL VELOCITY ASSIGNMENT'

DO 100 ITER=1,ITRN
CALL read_bfgs_file( pos, grad, fixion, energy, scratch, &
ni, stdout)
   DO I=1,N
       DO J=1,M
```

```
            A(J)=X(I,J)
            VI(J)=V(I,J)
        ENDDO
        CALL LSRCH(A,M,VI,NSTEPpso,FI)
        IF(FI.LT.F(I))THEN
        F(I)=FI
        DO IN=1,M
            BST(IN)=A(IN)
        ENDDO
        DO J=1,M
          XX(I,J)=A(J) !XX(I,J) is the M-tuple value of X
        ENDDO          !associated with local best F(I).
        ENDIF
        ENDDO

DO I=1,N
      BEST=1.0D30
      DO II=1,NN
          CALL RANDOM(RAND)
          NF=INT(RAND*N)+1
          IF(BEST.GT.F(NF)) THEN
          BEST=F(NF)
          NFBEST=NF
          ENDIF
      ENDDO
      DO J=1,M
          CALL RANDOM(RAND)
          V1(J)=A1*RAND*(XX(I,J)-X(I,J))
          CALL RANDOM(RAND)
          V2(J)=V(I,J)
          IF(F(NFBEST).LT.F(I)) THEN
          V2(J)=A2*W*RAND*(XX(NFBEST,J)-X(I,J))
          ENDIF
          V4(J)=W*V(I,J)
          V(I,J)=V1(J)+V2(J)+V4(J)
      ENDDO
      ENDDO

DO I=1,N
   DO J=1,M
   RANDS=0.D00
   X(I,J)=X(I,J)+V(I,J)*(1.D00+RANDS)
   ENDDO
ENDDO

DO I=1,N
   IF(F(I).LT.FMIN) THEN
   FMIN=F(I)
   II=I
   DO J=1,M
       BST(J)=XX(II,J)
   ENDDO
   ENDIF
ENDDO

IF(LCOUNT.EQ.NPRN) THEN
LCOUNT=0
WRITE(*,*)'OPTIMAL SOLUTION UP TO THIS POINT (FUNCTION
    CALLS=',NFCALL,')'
WRITE(*,*)'X = ',(BST(J),J=1,M),' MIN F = ',FMIN
WRITE(*,*)'CHECK SIZE OF X', SIZE(X(:,:))
IF(DABS(FFMIN-FMIN).LT.EPSILON) GOTO 999
FFMIN=FMIN
ENDIF
LCOUNT=LCOUNT+1

CALL write_bfgs_file( pos, energy, grad, scratch)
DO J=1,M
   POS(J)=BST(J)
ENDDO

100 CONTINUE
999 WRITE(*,*)'------------------------------------------'
WRITE(*,*)'FINAL X = ',(BST(J),J=1,M),' FINAL MIN F = ',FMIN
WRITE(*,*)'COMPUTATION OVER:FOR ',FTIT
WRITE(*,*)'NO. OF VARIABLES=',M,' END.'
WRITE(*,*)'NUMBER OF ITERATIONS USED', ITER


!------------------------------------------------------------
ENDIF

1000  CONTINUE
IF ( lmovecell ) FORALL( iii=1:3, ji=1:3) h(iii,ji) = pos(
     n-9 + ji+3*(iii-1) )
pos_in = pos(1:ni-9)
grad_in = grad(1:ni-9)

DEALLOCATE( pos )
DEALLOCATE( grad )
DEALLOCATE( pos_p )
DEALLOCATE( grad_p )
DEALLOCATE( pos_old )
DEALLOCATE( grad_old )
DEALLOCATE( inv_hess )
DEALLOCATE( step )
DEALLOCATE( step_old )
DEALLOCATE( pos_best )
DEALLOCATE( hinv_block )
DEALLOCATE( metric )

RETURN

CONTAINS

SUBROUTINE LSRCH(A,M,VI,NSTEPpso,FI)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
INTEGER :: M, NSTEPpso
REAL(DP) :: B(M), A(M), VI(MX)
M = SIZE(pos_in(:))
AMN=1.0D30
DO J=1,NSTEPpso
DO JJ=1,M
B(JJ)=A(JJ)+(J-(NSTEPpso/2)-1)*VI(JJ)
ENDDO
CALL FUNC(B,M,FI)
IF(FI.LT.AMN) THEN
AMN=FI
DO JJ=1,M
A(JJ)=B(JJ)
ENDDO
ENDIF
ENDDO
FI=AMN
RETURN
END SUBROUTINE LSRCH

SUBROUTINE RANDOM(RAND1)
DOUBLE PRECISION RAND1
COMMON /RNDM/IU,IV
INTEGER IU,IV
RAND=REAL(RAND1)
IV=IU*65539
IF(IV.LT.0) THEN
IV=IV+2147483547+1
ENDIF
RAND=IV
IU=IV
RAND=RAND*0.4656613E-09
RAND1=DBLE(RAND)
RETURN
END SUBROUTINE RANDOM

SUBROUTINE FSELECT(KF,M,FTIT)
```

```
INTEGER :: KF, M
CHARACTER *70 TIT(100),FTIT
DO I=1,91
WRITE(*,*)TIT(I)
ENDDO
KF=1
M=SIZE(POS(:))
FTIT=TIT(KF) !Stores the name of the chosen function in
             !FTIT.
RETURN
END SUBROUTINE FSELECT

SUBROUTINE FUNC(X,M,F)
INTEGER :: M, NFCALL, KF
COMMON /RNDM/IU,IV
COMMON /KFF/KF,NFCALL,FTIT
INTEGER IU,IV
REAL(DP) :: X, F, PI
DIMENSION X(*)
CHARACTER *70 FTIT
PI=4.D+00*DATAN(1.D+00)
NFCALL=NFCALL+1
IF(KF.EQ.1) THEN
F=energy
RETURN
ENDIF
STOP
END SUBROUTINE FUNC

END SUBROUTINE bfgs

SUBROUTINE read_bfgs_file( pos, grad, fixion, energy, &
                           scratch, ni, stdout )
IMPLICIT NONE
REAL(DP),         INTENT(INOUT) :: pos(:)
REAL(DP),         INTENT(INOUT) :: grad(:)
INTEGER,          INTENT(IN)    :: fixion(:)
CHARACTER(LEN=*), INTENT(IN)    :: scratch
INTEGER,          INTENT(IN)    :: ni
INTEGER,          INTENT(IN)    :: stdout
REAL(DP),         INTENT(INOUT) :: energy
CHARACTER(LEN=256) :: bfgs_file
LOGICAL            :: file_exists
REAL(DP) :: garbage, scnorm, trust_radius_ini
bfgs_file = TRIM( scratch ) // TRIM( prefix ) // '.bfgs'
INQUIRE( FILE = TRIM( bfgs_file ) , EXIST = file_exists )
IF ( file_exists ) THEN
OPEN( UNIT = iunbfgs, FILE = TRIM( bfgs_file ), &
STATUS = 'UNKNOWN', ACTION = 'READ' )
READ( iunbfgs, * ) pos_p
READ( iunbfgs, * ) grad_p
READ( iunbfgs, * ) scf_iter
READ( iunbfgs, * ) bfgs_iter
READ( iunbfgs, * ) gdiis_iter
READ( iunbfgs, * ) energy_p
READ( iunbfgs, * ) pos_old
READ( iunbfgs, * ) grad_old
READ( iunbfgs, * ) inv_hess
READ( iunbfgs, * ) tr_min_hit
CLOSE( UNIT = iunbfgs )
trust_radius_old = scnorm( pos(:) - pos_p(:) )
step_old = ( pos(:) - pos_p(:) ) / trust_radius_old
ELSE
WRITE( UNIT = stdout, FMT = '(/,5X,"BFGS Geometry
                                 Optimization")' )
call invmat(ni, metric, inv_hess, garbage)
pos_p      = 0.0_DP
grad_p     = 0.0_DP
scf_iter   = 0
bfgs_iter  = 0
gdiis_iter = 0
energy_p   = energy
step_old   = 0.0_DP
trust_radius_old = trust_radius_ini
pos_old  = 0.0_DP
grad_old = 0.0_DP
tr_min_hit = .FALSE.
END IF
END SUBROUTINE read_bfgs_file

SUBROUTINE write_bfgs_file( pos, energy, grad, scratch )
IMPLICIT NONE
REAL(DP),           INTENT(IN) :: pos(:)
REAL(DP),           INTENT(IN) :: energy
REAL(DP),           INTENT(IN) :: grad(:)
CHARACTER(LEN=*),   INTENT(IN) :: scratch
OPEN( UNIT = iunbfgs, FILE = TRIM( scratch )//TRIM( prefix
                                )//'.bfgs', &
STATUS = 'UNKNOWN', ACTION = 'WRITE' )
WRITE( iunbfgs, * ) pos
WRITE( iunbfgs, * ) grad
WRITE( iunbfgs, * ) scf_iter
WRITE( iunbfgs, * ) bfgs_iter
WRITE( iunbfgs, * ) gdiis_iter
WRITE( iunbfgs, * ) energy
WRITE( iunbfgs, * ) pos_old
WRITE( iunbfgs, * ) grad_old
WRITE( iunbfgs, * ) inv_hess
WRITE( iunbfgs, * ) tr_min_hit
CLOSE( UNIT = iunbfgs )
END SUBROUTINE write_bfgs_file

REAL(DP) FUNCTION scnorm( vect )
IMPLICIT NONE
REAL(DP), INTENT(IN) :: vect(:)
scnorm =SQRT( DOT_PRODUCT( vect , MATMUL( metric, vect ) ) )
END FUNCTION scnorm

SUBROUTINE terminate_bfgs( energy, energy_thr, grad_thr, &
cell_thr, lmovecell, stdout, scratch )
USE io_files, ONLY : prefix, delete_if_present
IMPLICIT NONE
REAL(DP),         INTENT(IN) :: energy, energy_thr, &
     grad_thr, cell_thr
LOGICAL,          INTENT(IN) :: lmovecell
INTEGER,          INTENT(IN) :: stdout
CHARACTER(LEN=*), INTENT(IN) :: scratch
IF ( conv_bfgs ) THEN
WRITE( UNIT = stdout, &
& FMT = '(/,5X,"bfgs converged in ",I3," scf cycles and ", &
&          I3," bfgs steps")' ) scf_iter, bfgs_iter
IF ( lmovecell ) THEN
WRITE( UNIT = stdout, &
& FMT = '(5X,"(criteria: energy < ",E8.2,", force < ",E8.2,
&       ", cell < ",E8.2,")")') energy_thr, grad_thr, &
&cell_thr
ELSE
WRITE( UNIT = stdout, &
& FMT = '(5X,"(criteria: energy < ",E8.2,", force < ",E8.2,
&                            ")")') energy_thr, grad_thr
END IF
WRITE( UNIT = stdout, &
& FMT = '(/,5X,"End of BFGS Geometry Optimization")' )
WRITE( UNIT = stdout, &
& FMT = '(/,5X,"Final ",A," = ",F18.10," Ry")' ) fname,
&energy
CALL delete_if_present( TRIM( scratch ) // TRIM( prefix ) //
                       '.bfgs' )
ELSE
WRITE( UNIT = stdout, &
FMT = '(/,5X,"The maximum number of steps has been
              reached.")' )
WRITE( UNIT = stdout, &
FMT = '(/,5X,"End of BFGS Geometry Optimization")' )
```

```
END IF
END SUBROUTINE terminate_bfgs


END MODULE bfgs_module
```

6.5 CORRESPONDENCE SEARCH ALGORITHM (FeTiH structure example)

```
% Rather than generating all the permutations for the arrangements, in this code
%two numbers are switched at a time and the new potential is compared.   Then
%a simulated annealing process is used to determine whether to accept or reject
%the change.

clear all;

x=2.956;         %Side lengths of FeTiH structure.
y=4.535;
z=4.388;
c=0.5;            %Constant to multiply with angular potential.
b=0.5;            %Constant to multiply with distance.
a=12;             %Number of planes in the structure for Fe atoms.
aTi=12;           %Number of planes in the structure for Ti atoms.

% v=[1:a];
% var=perms(v);

%Matlab does not allow perms(v) for large values of a (larger than 10?).
%Instead I will manually switch two of the numbers in v and compute the new
%potential and compare with the initial, then use delta/T criteria to
%accept or reject the change.


%The following are only for Fe planes.

p=[x/2,x,x/2,0,x/2,x/2,2*x,2*x,2*x,-x,-x,-x;          %Initial points for vectors in initial
    0,x/2,x,x/2,x/2,x/2,0,0,0,x,x,x;                   %state.
    x/2,x/2,x/2,x/2,x,0,0,0,0,x,x,x];

q=[y/2,y,y/2,0,y/2,y/2,y/2,y/2,y/2,y/2,y/2,y/2;     %Initial points for vectors in final
    0,z/2,z,z/2,z/2,z/2,z/2,z/2,z/2,z/2,z/2,z/2;     % state.
    x/2,x/2,x/2,x/2,x,0,x,x,x,0,0,0];

vp=[ 0,1,0,-1,0, 0,1,0, 0,1,0, 0;      %Direction vectors in initial state.
    -1,0,1, 0,0, 0,0,1,-1,0,1,-1;
     0,0,0, 0,1,-1,0,0, 0,0,0, 0;
```

```
        0,0,0, 0,0, 0,0,0, 0,0,0, 0];

vq=[0,1,0,-1,0, 0,1,0, 0,1,0, 0;      %Direction vectors in final state.
      -1,0,1, 0,0, 0,0,1,-1,0,1,-1;
        0,0,0, 0,1,-1,0,0, 0,0,0, 0;
        0,0,0, 0,0, 0,0,0, 0,0,0, 0];

%These are for the Ti planes.
pTi=[x/2,x/2,x/2,-x/2,-x/2,-x/2,x/2,x/2,x/2,3*(x/2),3*(x/2),3*(x/2);
        x/2,x/2,x/2,x/2,x/2,x/2,3*(x/2),3*(x/2),3*(x/2),x/2,x/2,x/2;
        x/2,x/2,x/2,x/2,x/2,x/2,x/2,x/2,x/2,x/2,x/2,x/2];
qTi=[x/2,x/2,x/2,0,0,0,x/2,x/2,x/2,x,x,x;
        0,0,0,x/2,x/2,x/2,x,x,x,x/2,x/2,x/2;
        x/2,x/2,x/2,x/2,x/2,x/2,x/2,x/2,x/2,x/2,x/2,x/2];
vpTi=[0,1,0,0,1,0,0,1,0,0,1,0;
         1,0,0,1,0,0,1,0,0,1,0,0;
         0,0,1,0,0,1,0,0,1,0,0,1];
vqTi=[0,1,0,0,1,0,0,1,0,0,1,0;
         1,0,0,1,0,0,1,0,0,1,0,0;
         0,0,1,0,0,1,0,0,1,0,0,1];

%Distance potential with final values set in the standard starting position.
for m=1:a
    initdist(1,m)=sqrt((q(1,m)-p(1,m))^2+(q(2,m)-p(2,m))^2+(q(3,m)-p(3,m))^2);
   initangle(1,m)=acos((dot(vp(:,m),vq(:,m)))/(sqrt((vp(1,m))^2+(vp(2,m))^2+(vp(3,
         m))^2)*(sqrt((vq(1,m))^2+(vq(2,m))^2+(vq(3,m))^2))));
end
for m=1:a
      initdistpotential(1,m)=b*(initdist(1,m))^2;
      initangularpotential(1,m)=c*(1-cos(initangle(1,m)));
end
initdistpotential=sum(initdistpotential);
initangularpotential=sum(initangularpotential);
initpotential=initdistpotential+initangularpotential;

%Same thing for the planes of the other species
for m=1:aTi
      initdistTi(1,m)=sqrt((qTi(1,m)-pTi(1,m))^2+(qTi(2,m)-pTi(2,m))^2+(qTi(3,m)-
       pTi(3,m))^2);
     initangleTi(1,m)=acos((dot(vpTi(:,m),vqTi(:,m)))/(sqrt((vpTi(1,m))^2+(vpTi(2,m)
       )^2+(vpTi(3,m))^2)*(sqrt((vqTi(1,m))^2+(vqTi(2,m))^2+(vqTi(3,m))^2))));
end
for m=1:aTi
      initdistpotentialTi(1,m)=b*(initdist(1,m))^2;
      initangularpotentialTi(1,m)=c*(1-cos(initangle(1,m)));
end
```

```
initdistpotentialTi=sum(initdistpotentialTi);
initangularpotentialTi=sum(initangularpotentialTi);
initpotentialTi=initdistpotentialTi+initangularpotentialTi;

%Swich two numbers in the arrangement and see if potential is reduced.
for h=0:5:500   %h represents gradual temperature change in 5 degree
    T=800-h;    %increments.
    i=randint(1,1,[1,a]);
    j=randint(1,1,[1,a]);
    newq=q;
    newqTi=qTi
    newq(:,i)=q(:,j);
    newq(:,j)=q(:,i);
    newqTi(:,i)=qTi(:,j);
    newqTi(:,j)=qTi(:,i);
    newvq=vq;
    newvqTi=vqTi;
    newvq(:,i)=vq(:,j);
    newvq(:,j)=vq(:,i);
    newvqTi(:,i)=vqTi(:,j);
    newvqTi(:,j)=vqTi(:,i);

    %Compute new potentials after switches are made.
    for m=1:a
        dist(1,m)=sqrt((newq(1,m)-p(1,m))^2+(newq(2,m)-p(2,m))^2+(newq(3,m)-
        p(3,m))^2);

        angle(1,m)=acos((dot(vp(:,m),newvq(:,m)))/(sqrt((vp(1,m))^2+(vp(2,m))^2
        +(vp(3,m))^2)*(sqrt((newvq(1,m))^2+(newvq(2,m))^2+(newvq(3,m))^2))));
    end
    for m=1:a
        distpotential(1,m)=b*(dist(1,m))^2;
        angularpotential(1,m)=c*(1-cos(angle(1,m)));
    end
    distpotential=sum(distpotential);
    angularpotential=sum(angularpotential);
    potential=distpotential+angularpotential;
    for m=1:aTi
        distTi(1,m)=sqrt((newqTi(1,m)-pTi(1,m))^2+(newqTi(2,m)-
        pTi(2,m))^2+(newqTi(3,m)-pTi(3,m))^2);

       angleTi(1,m)=acos((dot(vpTi(:,m),newvqTi(:,m)))/(sqrt((vpTi(1,m))^2+(vpTi(
       2,m))^2+(vpTi(3,m))^2)*(sqrt((newvqTi(1,m))^2+(newvqTi(2,m))^2+(newvq
       Ti(3,m))^2))));
    end
    for m=1:aTi
```

```
        distpotentialTi(1,m)=b*(distTi(1,m))^2;
        angularpotentialTi(1,m)=c*(1-cos(angleTi(1,m)));
    end
    distpotentialTi=sum(distpotentialTi);
    angularpotentialTi=sum(angularpotentialTi);
    potentialTi=distpotentialTi+angularpotentialTi;



    delta=(potential-initpotential)+(potentialTi-initpotentialTi);
    if delta<0
        q=newq;
        vq=newvq;
        initpotential=potential;
        qTi=newqTi;
        vqTi=newvqTi;
        initpotentialTi=potentialTi;
    elseif delta>0       %Still a chance to accept the change even if delta>0.
        r=rand(1);
        h=exp(-delta/T);
        if r<h
            q=newq;
            vq=newvq;
            initpotential=potential;
            qTi=newqTi;
            vqTi=newvqTi;
            initpotentialTi=potentialTi;
        end
    end
end

%Code for plane constraints. In this case planes 1 & 2 and 3 & 4 are used.
if (dot(newvq(:,1),newvq(:,2)) = dot(vq(:,1),vq(:,2))) &
  (dot(newvq(:,3),newvq(:,4)) = dot(vq(:,3),vq(:,4))) &
  (((p(:,1)-p(:,2))*vp(:,1))/(sqrt((vp(1,1))^2+(vp(2,1))^2+(vp(3,1))^2))) =
  (((q(:,1)-q(:,2))*vq(:,1))/(sqrt((vq(1,1))^2+(vq(2,1))^2+(vq(3,1))^2)))
  q=newq;
  vq=newvq;
  initpotential=potential;
end

vp
vq
vpTi
vqTi
initpotential;
```

```
qTi
vqTi
initpotentialTi;
totalpotential=initpotential+initpotentialTi
```

6.6 INTERMEDIATE POSITION SEARCH ALGORITHM WITH SURFACE LINEAR INTERPOLATION (FeTiH example)

```
% Find initial transition pathway of FeTi+H with simple nonlinear interpolation of
%basis vectors

clear all;

n=0.4           %Lambda value that is used in interpolation

isovalue1 = 0.0;
isovalue2 = 0;
isovalue3 = 0;

%Threshold values to find and plot atom positions.
isovalueFe = 0.0004;
isovalueTi = 0.0004;
isovalueH = 0.0004;

% Cell dimensions corresponding to Quantum-Espresso.
celldm = [5.58600098264628; 1.536874154262517; 1.484438430311231];

% Initial coordinates of atoms corresponding to Quantum-Espresso.
coordFe1_init = [0.000000000;    0.000000000;    0.000000000];
coordFe2_init = [0.000000000;    4.292490268;    0.000000000];
coordTi1_init = [2.793000491;    2.147048391;    4.146037265];
coordTi2_init = [2.793000491;    6.437932145;    4.146037265];
coordH1_init = [0.000000000;    0.000000000;    4.146037265];
coordH2_init = [0.000000000;    4.292490268;    4.146037265];

% Final coordinates of atoms corresponding to Quantum-Espresso.
coordFe1_final = [0.000000000;   -0.407605434;    0.000000000];
coordFe2_final = [0.000000000;    4.700081861;    4.146037265];
coordTi1_final = [2.793000491;   -0.104477173;    4.146037265];
coordTi2_final = [2.793000491;    4.396994224;    0.000000000];
coordH1_final = [0.000000000;    2.146238663;    2.072969363];
coordH2_final = [0.000000000;    2.146238663;    6.219105168];

% The actual size of the unit cell.
```

```
cellsize = [celldm(1); celldm(1)*celldm(2); celldm(1)*celldm(3) ];

% Basic x,y,z planes that intersect at (0,0,0)
A_x = [1 ];
H_x = [ 1;
          0 ;
          0 ];
alpha_x = [cellsize(1)/2];
lamda_x = [cellsize(1)*2];

A_y = [1 ];
H_y = [ 0 ;
          1 ;
          0 ];
alpha_y = [cellsize(2)/2];
lamda_y = [cellsize(2)*2];

A_z = [1 ];
H_z = [ 0 ;
          0 ;
          1];
alpha_z = [cellsize(3)/2];
lamda_z = [cellsize(3)*2];

 % Discretize the unit cell as a 3-D matrix
x=-0.0:cellsize(1)/100:cellsize(1);
y=-0.0:cellsize(2)/100:cellsize(2);
z=-0.0:cellsize(3)/100:cellsize(3);
[X,Y,Z]=meshgrid(x,y,z);

V = zeros(size(X,1),size(Y,1),size(Z,1));

%Defining planes and intersections of planes for each of the 14 points.
VxFe1trans = nodal(A_x, H_x, (1-n)*(alpha_x+coordFe1_init(1))+n*
         (alpha_x+coordFe1_final(1)), lamda_x, X,Y,Z);
VyFe1trans = nodal(A_y, H_y, (1-n)*(alpha_y+coordFe1_init(2))+n*
         (alpha_y+coordFe1_final(2)), lamda_y, X,Y,Z);
VzFe1trans = nodal(A_z, H_z, (1-n)*(alpha_z+coordFe1_init(3))+n*
         (alpha_z+coordFe1_final(3)), lamda_z, X,Y,Z);

VxFe2trans = VxFe1trans;     %% Shared plane
VyFe2trans = nodal(A_y, H_y, (1-n)*(alpha_y+coordFe2_init(2))+n*
         (alpha_y+coordFe2_final(2)), lamda_y, X,Y,Z);
VzFe2trans = nodal(A_z, H_z, (1-n)*(alpha_z+coordFe2_init(3))+n*
         (alpha_z+coordFe2_final(3)), lamda_z, X,Y,Z);
```

```
VxTi1trans = nodal(A_x, H_x, (1-n)*(alpha_x+coordTi1_init(1))+n*
        (alpha_x+coordTi1_final(1)), lamda_x, X,Y,Z);
VyTi1trans = nodal(A_y, H_y, (1-n)*(alpha_y+coordTi1_init(2))+n*
        (alpha_y+coordTi1_final(2)), lamda_y, X,Y,Z);
VzTi1trans = nodal(A_z, H_z, (1-n)*(alpha_z+coordTi1_init(3))+n*
        (alpha_z+coordTi1_final(3)), lamda_z, X,Y,Z);

VxTi2trans = VxTi1trans;    %% Shared plane
VyTi2trans = nodal(A_y, H_y, (1-n)*(alpha_y+coordTi2_init(2))+n*
        (alpha_y+coordTi2_final(2)), lamda_y, X,Y,Z);
VzTi2trans = nodal(A_z, H_z, (1-n)*(alpha_z+coordTi2_init(3))+n*
        (alpha_z+coordTi2_final(3)), lamda_z, X,Y,Z);

VxH1trans = nodal(A_x, H_x, (1-n)*(alpha_x+coordH1_init(1))+n*
        (alpha_x+coordH1_final(1)), lamda_x, X,Y,Z);
VyH1trans = nodal(A_y, H_y, (1-n)*(alpha_y+coordH1_init(2))+n*
        (alpha_y+coordH1_final(2)), lamda_y, X,Y,Z);
VzH1trans = nodal(A_x, H_z, (1-n)*(alpha_z+coordH1_init(3))+n*
        (alpha_z+coordH1_final(3)), lamda_z, X,Y,Z);

VxH2trans = VxH1trans;   %% Shared plane
VyH2trans = nodal(A_y, H_y, (1-n)*(alpha_y+coordH2_init(2))+n*
        (alpha_y+coordH2_final(2)), lamda_y, X,Y,Z);
VzH2trans = nodal(A_z, H_z, (1-n)*(alpha_z+coordH2_init(3))+n*
        (alpha_z+coordH2_final(3)), lamda_z, X,Y,Z);

Int1 = VxFe1trans.^2+VyFe1trans.^2+VzFe1trans.^2;    %%%Fe1
Int2 = VxFe2trans.^2+VyFe2trans.^2+VzFe2trans.^2;    %%%Fe2

Int3 = VxTi1trans.^2+VyTi1trans.^2+VzTi1trans.^2;    %%%Ti1
Int4 = VxTi2trans.^2+VyTi2trans.^2+VzTi2trans.^2;    %%%Ti2

Int5 = VxH1trans.^2+VyH1trans.^2+VzH1trans.^2;    %%%H1
Int6 = VxH2trans.^2+VyH2trans.^2+VzH2trans.^2;    %%%H2

%Show marker at each intersection.

figure('Color','white'); hold on;

S = size(Int1);
V=zeros(S);
for i=1:S(1)
    for j=1:S(2)
        for k=1:S(3)
            if Int1(i,j,k)<=isovalueFe
                plot3(X(i,j,k),Y(i,j,k),Z(i,j,k),'r.','MarkerSize',240);
                Fe1 = [X(i,j,k); Y(i,j,k); Z(i,j,k)]
            end
        end
    end
end

S = size(Int2);
V=zeros(S);
for i=1:S(1)
    for j=1:S(2)
        for k=1:S(3)
            if Int2(i,j,k)<=isovalueFe
                plot3(X(i,j,k),Y(i,j,k),Z(i,j,k),'r.','MarkerSize',240);
                Fe2 = [X(i,j,k); Y(i,j,k); Z(i,j,k)]
            end
        end
    end
end

S = size(Int3);
V=zeros(S);
for i=1:S(1)
    for j=1:S(2)
        for k=1:S(3)
            if Int3(i,j,k)<=isovalueTi
                plot3(X(i,j,k),Y(i,j,k),Z(i,j,k),'.','Color',[0 1 0],'MarkerSize',240);
                Ti1 = [X(i,j,k); Y(i,j,k); Z(i,j,k)]
            end
        end
    end
end

S = size(Int4);
V=zeros(S);
for i=1:S(1)
    for j=1:S(2)
        for k=1:S(3)
            if Int4(i,j,k)<=isovalueTi
                plot3(X(i,j,k),Y(i,j,k),Z(i,j,k),'.','Color',[0 1 0],'MarkerSize',240);
                Ti2 = [X(i,j,k); Y(i,j,k); Z(i,j,k)]
            end
        end
    end
end

S = size(Int5);
```

```
V=zeros(S);
for i=1:S(1)
      for j=1:S(2)
            for k=1:S(3)
                  if Int5(i,j,k)<=isovalueH
                        plot3(X(i,j,k),Y(i,j,k),Z(i,j,k),'c.','MarkerSize',80);
                        H1 = [X(i,j,k); Y(i,j,k); Z(i,j,k)]
                  end
            end
      end
end

S = size(Int6);
V=zeros(S);
for i=1:S(1)
      for j=1:S(2)
            for k=1:S(3)
                  if Int6(i,j,k)<=isovalueH
                        plot3(X(i,j,k),Y(i,j,k),Z(i,j,k),'c.','MarkerSize',80);
                        H2 = [X(i,j,k); Y(i,j,k); Z(i,j,k)]
                  end
            end
      end
end




draw_ps(X,Y,Z,VxFe1trans,isovalue1,0.1,'yellow');
draw_ps(X,Y,Z,VyFe1trans,isovalue2,0.1,'blue');
draw_ps(X,Y,Z,VzFe1trans,isovalue3,0.1,'green');
%
draw_ps(X,Y,Z,VyFe2trans,isovalue2,0.1,'blue');
%
draw_ps(X,Y,Z,VyH1trans,isovalue2,0.1,'yellow');
%
draw_ps(X,Y,Z,VyH2trans,isovalue2,0.1,'yellow');

axis equal;
view([51,17]);
hold off;
```

6.7 INITIAL AND FINAL POSITIONS AND CELL DIMENSIONS (FePt)

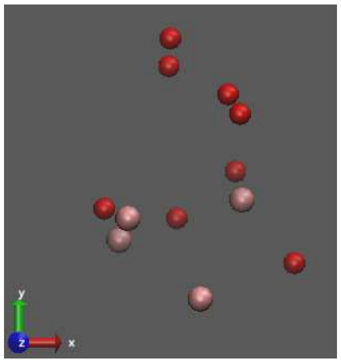
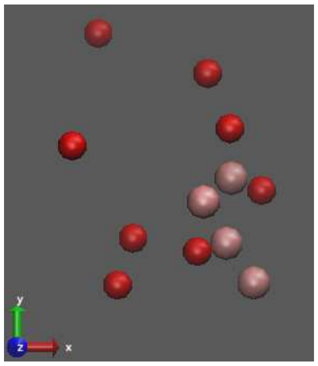
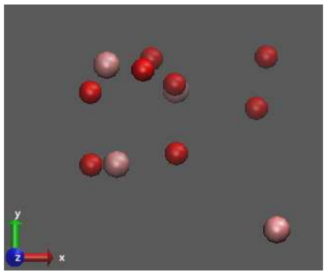
```
%%% Initial coordinates of atoms corresponding to Quantum-Espresso.
coordFe1_init = [3.660380211,      0.000000000,      3.509202918];
coordFe2_init = [3.660380211,      3.660380211,      0.000000000];
coordPt1_init = [0.000000000,      0.000000000,      0.000000000];
coordPt2_init = [0.000000000,      3.660380211,      3.509202918];

%%% Final coordinates of atoms corresponding to Quantum-Espresso.
coordFe1_final = [3.660380211,      0.000000000,      3.509202918];
coordFe2_final = [0.000000000,      3.660380211,      3.509202918];
coordPt1_final = [0.000000000,      0.000000000,      0.000000000];
coordPt2_final = [3.660380211,      3.660380211,      0.000000000];

%%% The actual size of the unit cell.
celldm = [7.32076042177662; 1; 0.958699019101704];
cellsize = [celldm(1); celldm(1)*celldm(2); celldm(1)*celldm(3) ];
```

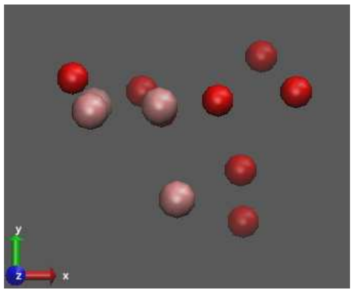
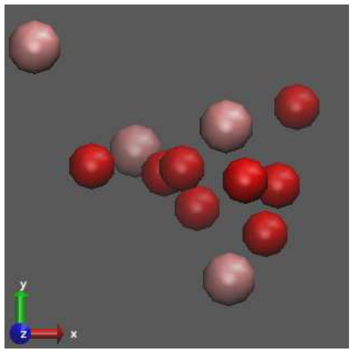
6.8 INITIAL AND FINAL POSITIONS AND CELL DIMENSIONS (VO_2)

```
%%% Initial coordinates of atoms corresponding to Quantum-Espresso.
coordV1_init = [0.000000000     0.000000000     0.000000000];
coordV2_init = [4.303450622     4.303450622     2.694165438];
coordV3_init = [0.000000000     0.000000000     5.388336672];
coordV4_init = [4.303450622     4.303450622     8.082507906];
coordO1_init = [1.724380595     1.724380595     2.694167742];
coordO2_init = [6.882520648     6.882520648     2.694167742];
coordO3_init = [6.028197539     2.578703704     0.000000000];
coordO4_init = [2.578703704     6.028197539     0.000000000];
coordO5_init = [1.724380595     1.724380595     8.082505602];
coordO6_init = [6.882520648     6.882520648     8.082505602];
coordO7_init = [6.028197822     2.578703421     5.388336672];
coordO8_init = [2.578703421     6.028197822     5.388336672];

%%% Final coordinates of atoms corresponding to Quantum-Espresso.
coordV1_final = [-0.480280207     0.067047838     0.000000000];
coordV2_final = [4.538374156     4.743830495     2.760782806];
coordV3_final = [-0.001176736     0.655526527     5.657753506];
coordV4_final = [4.538374156     4.743830495     8.554724205];
coordO1_final = [1.692399181     1.801384761     2.958551401];
coordO2_final = [7.224617185     7.318169591     2.633469670];
coordO3_final = [6.297190733     2.742166323     0.000000000];
coordO4_final = [2.890028988     6.142496562     0.000000000];
coordO5_final = [1.692399181     1.801384761     8.356954986];
coordO6_final = [7.224617185     7.318169591     8.682036717];
```

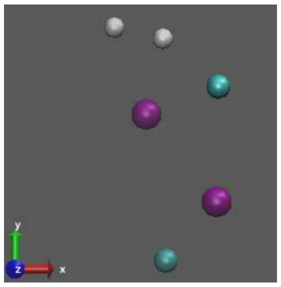
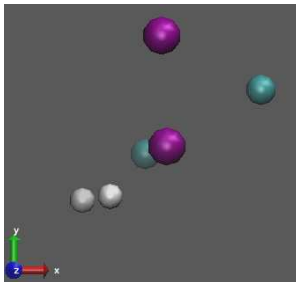
```
coordO7_final = [6.420737676    2.739544822    5.657753506];
coordO8_final = [3.148950028    6.110294353    5.657753506];

%%% The actual size of the unit cell.
cellsize = [9.037246305, 9.037246305, 11.315507011 ];
```

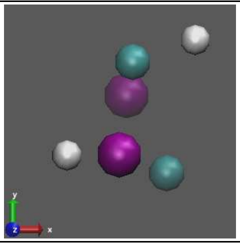
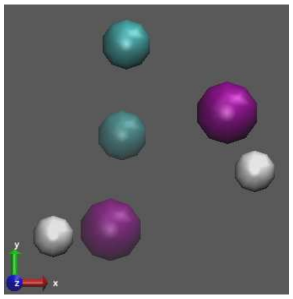
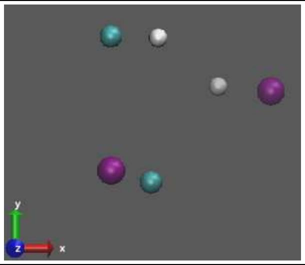
6.9 VO_2 RELAXATION RESULTS WITH RG-PSO

Structure	Energy	Positions
	-366.019	4.423980941, 1.671226267, 3.979258887 0.881107209, 0.514755910, 2.290170453 3.225147374, 1.178981302, 6.731167542 1.125862848, 1.143979101, 5.839797111 4.229051123, 2.498240801, 1.117036941 4.020334630, 4.705744403, 4.998290693 5.930616251, 0.151394508, 3.365915590 2.352194655, 6.320788947, 3.982557505 4.360103321, 4.149141849, 5.417612214 0.445131877, 1.421900419, 3.445483234 2.540591195, 1.141025929, 1.347512024 2.309596249, 5.520135246, 3.393616808
	-365.205	3.512396613, 2.501422437, 1.568790528 3.408344381, 1.070480545, 2.571516979 2.907960065, 1.986159632, 3.617672413 4.009659454, 0.203301767, 2.228267432 4.176966829, 2.232262846, 1.516185974 2.778613200, 0.898073397, 2.638325822 3.485321237, 3.590923680, 3.174748326 1.366347059, 1.186188780, 2.088690271 0.033731880, 3.216409133, 4.629438456 0.600199617, 5.685626781, 0.338921628 1.022233461, 0.152346475, 2.639870101 2.994653483, 4.803863405, 1.908338800
	-364.581	1.162002535, 2.344680951, 3.355575741 2.777390612, 4.364518664, 0.227653379 5.509609097, 0.575114951, 6.461794239 0.899258598, 5.033635656, 4.251383044 2.781605042, 2.637730563, 2.896756192 5.224550779, 5.251772380, 0.835794261 0.470561368, 2.320449483, 2.600190471 4.954059738, 3.864296165, 0.767016901 0.451955436, 4.294783951, 4.134775105 2.741926466, 4.516266061, 1.568480890 1.888702781, 4.907534556, 5.453433397 2.107891842, 5.203145890, 1.048401156

	-364.462	0.638363732, 2.783124044, 3.930149240 0.710928099, 2.882327841, 0.290671868 2.354330648, 1.030716975, 3.584532472 2.008385886, 2.883097743, 3.411364708 3.612912052, 1.617209609, 1.863968747 2.041891527, 2.780039448, 2.744831804 1.664561572, 3.178170522, 1.042970273 3.164370664, 2.995092972, 0.051480363 0.296276588, 3.434062812, 4.835760551 4.024220886, 3.864578350, 1.199596073 4.713999796, 3.167505020, 4.387928625 3.665317670, 0.591251998, 0.620349980
	-362.093	0.857343480, 4.433694530, 2.777516367 3.491745848, 3.337652723, 2.466042776 2.249927120, 2.993231547, 0.914456659 3.535278004, 1.210399939, 1.945216088 1.643026696, 2.780510091, 2.244532082 2.647269830, 2.681438296, 1.384082198 3.756313192, 2.583090549, 3.348028584 3.091840960, 2.201913476, 0.992015042 4.465527069, 3.603190365, 1.145714824 2.884348880, 2.755038177, 1.688941206 4.201689084, 2.515373460, 2.294839157 4.039091386, 1.841373219, 2.034115517

6.10 FeTiH RELAXATION RESULTS WITH RG-PSO

Structure	Energy	Positions
	-346.731	4.571887260, 3.555060352, 5.904005979 3.320016284, 0.600950297, 2.086143099 2.848045363, 2.887198547, 1.608489132 4.528391179, 0.803799518, 2.631356446 3.242845557, 4.691790913, 3.501430909 2.084460989, 4.957379136, 3.055452812
	-346.529705	4.642583860, 3.067931619, 3.024552226 2.384445372, 1.809753390, 0.597041077 2.801944018, 1.952110752, 4.001081126 2.707209123, 4.120775042, 4.535899024 1.151817552, 0.895773520, 2.915068288 1.690803253, 0.970809589, 4.716733135

	-346.529704	2.770318295, 3.962055301, 1.768692071 3.334395492, 2.148204458, 0.959363678 2.672309576, 3.418875375, 0.377336117 2.552527428, 2.441578643, 3.210695998 1.694561953, 2.435370780, 3.299105176 3.804746124, 4.328960901, 3.375027624
	-346.528	2.335825878, 5.327399190, 3.310146107 2.289501287, 4.236778767, 1.751725994 3.543183724, 4.505598791, 3.516631183 2.143349208, 3.107171495, 1.920265024 1.450201587, 3.029518746, 3.771372001 3.879081650, 3.806753482, 3.928010924
	-346.529	2.155944831, 0.953057993, 2.319383952 1.111230488, 4.725245291, 3.410717186 5.260873796, 3.300891121, 0.190687058 1.129042926, 1.256196117, 1.653106099 2.344451609, 4.680518576, 6.207635787 3.891310758, 3.424067911, 1.530297612

6.11 RELAXATION RESULTS WITH RG-PSO USING FORCED INITIAL POSITIONS

FeTiH Structure:

Positions	Energy
0.000000000, 0.000000000, 0.000000000 0.000000000, 4.292490482, 0.000000000 2.793000459, 2.146245241, 4.146037101 2.793000459, 6.438735485, 4.146037101 0.000000000, 0.000000000, 4.146037101 0.000000000, 4.292490482, 4.146037101	-346.52938
1.216259504, 4.659608126, 1.289208427 4.431491242, 2.365923136, 2.777527199 1.972322458, 2.852667117, 3.747156084 0.409371907, 4.181063469, 2.157675349 1.869283166, 2.973086544, 5.228643027 1.389285094, 1.691310855, 0.064644718	-346.52941
5.114182636, 3.825653589, 2.508126871 0.031724322, 1.960965000, 3.013705387 4.997944691, 3.294378376, 1.134801894 1.024940564, 1.401822718, 0.315897688 4.594663003, 2.303588045, 4.146407279	-346.5285

3.832872406, 4.067084974, 2.807462293	
0.270155079, 1.696123881, 2.187758676 3.892564138, 2.638100793, 1.797057248 2.541189487, 2.037428607, 2.938413414 1.204724212, 5.049720549, 3.447798687 3.319753923, 3.304638466, 4.325110963 3.115009714, 3.867357963, 2.409619527	-346.5216
1.036358760, 4.231627832, 3.219428045 2.102036586, 4.078738570, 1.211394200 4.149277691, 1.835658385, 4.290093443 2.169903538, 3.946845024, 3.860114766 2.733948222, 1.982080847, 3.495538518 4.643507954, 4.413042142, 2.678111030	-346.5215

BFGS searching result:

Positions:	0.000000000	0.000000000	0.000000000
	0.000000000	4.292490268	0.000000000
	2.793000491	2.147048391	4.146037265
	2.793000491	6.437932145	4.146037265
	0.000000000	0.000000000	4.146037265
	0.000000000	4.292490268	4.146037265

Energy: -346.529387

VO_2 Structure:

Positions	Energy
0.000000000, 0.000000000, 0.000000000 4.518623415, 4.518623532, 2.828850156 0.000000000, 0.000000000, 5.657753523 4.518623415, 4.518623415, 8.486656422 1.728094242, 1.728094242, 2.828853424 7.309151619, 7.309151619, 2.828853252 6.247159185, 2.790087120, 0.000000000 2.790087120, 6.247159185, 0.000000000 1.728094686, 1.728094686, 8.486653759 7.309151619, 7.309151619, 8.486653759 6.247165118, 2.790081187, 5.657753506 2.790081187, 6.247165118, 5.657753506	-367.94
2.015851166, 2.672613379, 1.301796643 4.152902573, 1.543624201, 4.323921905 1.478822024, 1.345980500, 2.692661580 3.262738863, 4.547325571, 3.036211701 2.431135820, 1.022824891, 3.024056946 2.682022305, 2.982274864, 3.399130587 0.375979038, 0.140044109, 4.125434159 4.081966683, 1.917919506, 3.661266396 1.588261658, 2.340935221, 3.944295105 3.608907577, 2.033204320, 4.386557394 1.817631559, 1.309084864, 0.054206631	-366.0195

1.135938180, 1.210379768, 0.570777573	
4.404177963, 2.883651064, 1.737073929 0.924983712, 0.632543840, 2.239561067 1.660487349, 1.082059216, 0.592040399 4.861675158, 3.623126750, 1.634677641 2.081444345, 1.649563076, 3.405428381 2.782590779, 1.753521253, 2.225375825 3.92043959, 0.5781382493, 4.237783565 3.044125541, 0.373883516, 3.296594040 1.519057883, 2.462209468, 3.869328435 2.018888996, 3.175352762, 3.012880256 2.960987108, 2.781104619, 1.571398896 0.555315760, 2.312509271, 0.064801085	-364.6211
3.592929039, 1.841863884, 3.710910789 3.901808355, 0.567201937, 4.042351040 3.812865306, 3.407608548, 2.476622619 0.217116252, 1.436856073, 3.270920147 3.531734724, 1.479858627, 4.839454249 2.284658562, 5.308289635, 3.651016468 2.309321798, 2.257862768, 5.033363483 3.683478871, 1.556613318, 1.695186205 5.157794989, 0.2636789301, 4.791443712 1.9183238074, 1.675960390, 4.717836509 0.7455009915, 2.012157067, 2.331880854 2.105525298, 1.658401879, 0.3153178238	-365.9589
4.393766816, 1.706443327, 3.667199613 0.515105675, 0.185985287, 2.253273223 3.113785912, 1.854231101, 0.150070561 5.550261402, 2.941201871, 0.527724947 5.786992793, 1.888754457, 2.458672167 0.124839535, 3.104469597, 3.256454805 6.453235694, 5.496081131, 4.658800366 2.181798558, 1.892027131, 4.736524186 3.772382210, 1.923715865, 3.812733253 2.085740065, 0.244363281, 4.334670500 4.223438323, 2.689116862, 3.314449409 1.383127876, 4.349482959, 0.810463894	-363.4634

BFGS searching result:

Positions:	0.000000000	0.000000000	0.000000000
	4.518623153	4.518623153	2.828850173
	0.000000000	0.000000000	5.657753506
	4.518623153	4.518623153	8.486656838
	1.728094686	1.728094686	2.828853252
	7.309151619	7.309151619	2.828853252
	6.247159185	2.790087120	0.000000000
	2.790087120	6.247159185	0.000000000
	1.728094686	1.728094686	8.486653759
	7.309151619	7.309151619	8.486653759
	6.247165118	2.790081187	5.657753506
	2.790081187	6.247165118	5.657753506

Energy: -367.942756

7. REFERENCES

1 Yeomans J.M. *Statistical Mechanics of Phase Transitions*, 2002; New York: Oxford University Press

2 Papon P., Leblond J., and Meijer P.H.E. *The Physics of Phase Transitions: Concepts and Applications*, 2006; Berlin: Springer-Verlag

3 Lavrov A.N., Komiya S., and Ando Y. Antiferromagnets: Magnetic shape-memory effects in a crystal. *Nature*, **418**: 385-386

4 Mnyukh Y. *Fundamentals of Solid-State Phase Transitions, Ferromagnetism and Ferroelectricity*, 1st Books Library 2001.

5 Cerjan C., and Miller W. On finding transition states. *J. Chem. Phys.* 1981; **75**: 2800-2806.

6 Dewar, M.J.S., Healy, E.F., and Stewart, J.J.P. Location of transition states in reaction mechanisms. *J. Chem. Soc., Faraday Trans.* 1984; **2**(80): 52-57.

7 Wang Y. Periodic surface modeling for computer aided nano design. *Computer-Aided Design* 2007; **39**(3): 179-189.

8 Qi C. and Wang Y. Feature-Based Crystal Construction with Periodic Surfaces. *Computer-Aided Design* 2009; **41**(11): 792-800.

9 Wang Y. Loci periodic surface reconstruction from crystals. *Computer-Aided Design & Applications* 2007; **4**(1-4): 437-447.

10 Wang Y. Degree elevation and reduction of periodic surfaces. *Computer-Aided Design & Applications* 2008; **5**(6): 841-854.

11 Wang Y. Computing Minkowski sums of periodic surface models. *Computer-Aided Design & Applications* 2009; **6**(6): 825-837.

12 Edelsbrunner H. Deformable smooth surface design. *Discrete & Computational Geometry* 1999; **21**(1): 87-115.

13 Edelsbrunner H. Surface reconstruction by wrapping finite point sets in space. *Discrete and Computation Geometry. The Goodman-Pollack Festschrift* 2003; 379-404.

14 Bajaj C., Pascucci V., Shamir A., Holt R., and Netravali A. Dynamic maintenance and visualization of molecular surfaces. *Discrete Applied Mathematics* 2003; **127**(1): 23-51

15 Zhang Y., Xu G., and Bajaj C. Quality meshing of implicit solvation models of biomolecular structures. *Computer-Aided Geometric Design* 2006; **23**(6):510-530.

16 Ryu J., Park R., and Kim D.-S. Molecular surfaces on proteins via beta shapes. *Computer Aided Design* 2007; **39**(12): 1042-1057.

17 Ryu J., Cho Y., and Kim D.-S. Triangulation of molecular surfaces. *Computer Aided Design* 2009; **41**(6): 463-478.

18 Kim D.-S., Cho Y., Sugihara K., Ryu J., and Kim D. Three-dimensional beta-shapes and beta-complexes via quasi-triangulation. *Computer Aided Design* 2010; **42**(10): 911-929.

19 Au C.K. and Woo T.C. Ribbons: Their geometry and topology. *Computer-Aided Design & Applications* 2004; **1**(1-4):1-6.

20 Kim D.-S., Cho Y., and Kim D. Euclidean Voronoi diagram of 3D balls and its computation via tracing edges. *Computer-Aided Design* 2005; **37**(13):1412-1424.

21 Bajaj, C. A Laguerre Voronoi based scheme for meshing particle swarms. *Japan Journal of Industrial and Applied Mathematics* 2005; **22**(2): 167-177.

22 Jonsson, H., Mills, G., and Jacobsen, K. 1998. *Classical and Quantum Dynamics in Condensed Phase Simulations.* World Scientific, Hackensack, NJ, Chap. 16, pp. 385-404.

23 Henkelman, G., and Jonsson, H. Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points. *J. Chem. Phys.* 2000; **113**(22): 9978-9985.

24 Trygubenko, S., and Wales, D. A doubly nudged elastic band method for finding transition states. *J. Chem. Phys.* 2004; **120**(5): 2082-2094.

25 Zhu, T., Li, J., Samanta, A., Kim, H.G., and Suresh, S. Interfacial plasticity governs strain rate sensitivity and ductility in nanostructured metals. *Proceedings of the National Academy of Sciences of the USA* 2007; **104**(9): 3031-3036.

26 Henkelman G., Uberuaga B.P., and Jonsson H. A climbing image nudged elastic band method for finding saddle points and minimum energy paths. *Journal of Chemical Physics* 2000; **113**(22): 9901-9904.

27 Galvan I., and Field M. Improving the Efficiency of the NEB Reaction Path Finding Algorithm. *J. Comp. Chem.* 2008; **29**(1): 139-143.

28 E W., Ren W., and Vanden-Ejinden E. String method for the study of rare events. *Phys. Rev. B*, 2002; **66**(5): 052301(1-4).

29 Ren W. Higher order string method for finding minimum energy path. *Comm. Math. Sci.*, 2003; **1**(2): 377-384.

30 Peters B., Heyden A., Bell A.T., and Chakraborty A. A growing string method for determining transitions states: Comparison to the nudged elastic band and string methods. *J. Chem. Phys.* 2004; **120**(17): 7877-7886.

31 Burger S., and Yang W. Quadratic string method for determining the minimum-energy path based on multiobjective optimization. *J. Chem. Phys.* 2006; **124**(5): 054109(1)-054109(12).

32 Chen L., Ying C., and Ala-Nissila T. Finding transition paths and rate coefficients through accelerated Langevin dynamics. *Phys. Rev. E.* 2002; **65**(4): 042101(1)-042101(4).

33 Fischer S., and Karplus M. Conjugate Peak Refinement: an algorithm for finding reaction paths and accurate transition states with many degrees of freedom. *Chem. Phys. Lett.* 1992; **194**(3): 252-261.

34 Dey B., and Ayers P. A Hamilton-Jacobi type equation for computing minimum potential energy paths. *Mol. Phys.* 2006; **104**(4): 541-558.

35 Simons J., Joergensen P., Taylor, H., Ozment, J. Walking on Potential Energy Surfaces. *J. Phys. Chem.* 1983; **87**(15): 2745-2753.

36 Banerjee A., Adams N., Simons J., and Shepard R. Search for Stationary Points on Surfaces. *J. Phys. Chem.* 1985; **89**(1): 52-57.

37 Ionova I.V. and Carter E.A. Ridge method for finding saddle points on potential energy surfaces. *J. Chem. Phys.* 1993; **98**(8): 6377-6386.

38 Henkelman G. and Jonsson H. A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives. *J. Chem. Phys.* 1999; **111**(15): 7010-7022.

39 Quapp W., Hirsch M., Imig O., and Heidrich D. Searching for Saddle Points of a Potential Energy Surface by Following a Reduced Gradient. *J. Comput. Chem.* 1998; **19**(9): 1087-1100.

40 Anglada J.M., Besalu E., Bofill J.M., and Crehuet R. On the Quadratic Reaction Path Evaluated in a Reduced Potential Energy Surface Model and the Problem to locate Transition States. *J. Comput. Chem.* 2001; **22**(4): 387-406.

41 Govind N., Petersen M., Fitzgerald G., King-Smith D., and Andzelm J. A generalized synchronous transit method for transition state location. *Computational Materials Science* 2003; **28**(2): 250-258.

42 Dewar M.J.S., Healy E.F., and Stewart J.J.P. Location of Transition States in Reaction Mechanisms. *J. Chem. Soc., Faraday Trans.* 1984; **2**(80): 227.

43 Mousseau N. and Barkema G.T. Traveling through potential energy landscapes of disordered materials: The activation-relaxation technique. *Physical Review E* 1998; **57**(2): 2419-2424.

44 Miron R.A. and Fichthorn K.A. The Step and Slide method for finding saddle points on multidimensional potential surfaces. *J. Chem. Physics* 2001; **115**(19): 8742-8747.

45 Lin Y. and Stadtherr M.A. Locating stationary points of Sorbate-Zeolite potential energy surfaces using interval analysis. *J. Chem. Phys.* 2004; **121**(20): 10159-10166.

46 Kennedy J. and Eberhart R. Particle Swarm Optimization. *Neural Networks* 1995; **4**: 1942-1948.

47 Blackwell T.M. and Bentley P. Don't Push Me! Collision-Avoiding Swarms. *Evolutionary Computation* 2002; 1691-1696.

48 Lovbjerg M. and Krink T. Extending Particle Swarm Optimisers with Self-Organized Criticality. *Evolutionary Computation* 2002; 1588-1593.

49 Xie X.-F., Zhang W.-J., and Yang Z.-L. Dissipative Particle Swarm Optimization. *Evolutionary Computation* 2002; 1456-1461.

50 Liang, J.J. and Suganthan, P.N. Dynamic Multi-Swarm Particle Swarm Optimizer. *Swarm Intelligence Symposium* 2005; 124-129.

51 Zhao, S.Z., Liang, J.J., Suganthan, P.N., and Tasgetiren, M.F. Dynamic Multi-Swarm Particle Optimizer with Local Search for Large Scale Global Optimization. *Evolutionary Computation* 2008; 3845-3852.

52 Janson, S. and Middendorf, M. A Hierarchical Particle Swarm Optimizer and Its Adaptive Variant. *Systems, Man, and Cybernetics* 2005; **35**(6): 1272-1282.

53 Iqbal, M. and Montes de Oca, M.A. An Estimation of Distribution Particle Swarm Optimization Algorithm. *Lecture Notes in Computer Science* 2006; **4150**: 72-83.

54 Krink, T., Vesterstrom, J.S., and Riget, J. Particle Swarm Optimisation With Spatial Particle Extension. *Evolutionary Computation* 2002; 1474-1479.

55 Mishra, S.K., Performance of Repulsive Particle Swarm Method in Global Optimization of Some Important Test Functions: A Fortran Program. *Social Science Research Network.* 2006.

56 Kinaci A. and Aydinol M.K. Ab initio investigation of FeTi-H system. *International Journal of Hydrogen Energy* 2007; **32**(13): 2466-2474.

57 Giannozzi P., Baroni, S., Bonini, N., Calandra, M., Car, R., Cavazzoni, C., Ceresoli, D., Chiarotti, G., Cococcioni, M., Dabo, I., Dal Corso, A., Fabris, S., Fratesi, G., De Gironcoli, S., Gebauer, R., Gerstmann, U., Gougoussis, C., Kokalj, A., Lazzeri, M., Martin-Samos, L., Marzari, N., Mauri, F., Mazzarello, R., Paolini, S., Pasquarello, A., Paulatto, L., Sbraccia, C., Scandolo, S., Sclauzero, G., Seitsonen, A.P., Smogunov, A., Umari, P., and Wentzcovitch, R.M. Quantum Espresso: a modular and open-source software project for quantum simulations of materials. *Journal of Physics: Condensed Matter* 2009; **21**(39): 395502(1-19).

58 Izanlou, A., Aydinol, M.K. An ab initio study of dissociative adsorption of H2 on FeTi surfaces. *International Journal of Hydrogen Energy* 2010; **35**(4), 1681-1692.

59 Sen, F.G., Kinaci, A., Aydinol, M.K. Effect of alloying elements on the formation of FeTiH: An ab initio study. *Carbon Nanomaterials in Clean Energy Hydrogen Systems*, 2009: 573-578.

60 Bogicevic, A. Hyldgaard, P., Wahnstrom, G., and Lundqvist, B. Al Dimer Dynamics on Al(111). *Physical Review Letters* 1998; **81**(1): 172-175.

61 Stumpf, R. and Scheffler, M. Theory of Self-Diffusion at and Growth of Al(111). *Physical Review Letters* 1993; **72**(2): 254-257.

62 Berglund, C.N. and Guggenheim, H.J. Electronic Properties of VO_2 near the Semiconductor-Metal Transition. *Physical Review* 1969; **185**(3): 1022-1033.

63 Barmak, K., Kim, J., Lewis, L.H., Coffey, K.R., Toney, M.F., Kellock, A.J. and Thiele, J.-U.. On the relationship of magnetocrystalline anisotropy and stoichiometry in epitaxial $L1_0$ CoPt (001) and FePt (001) thin films. *Journal of Applied Physics* 2005; **98**(3): 033904(1-10).

64 Barmak, K., Kim, J., Shell, S., Svedberg, E.V. and Howard, J.K. Calorimetric studies of the A1 to $L1_0$ transformation in FePt and CoPt thin films. *Applied Physics Letters* 2002, **80**(22): 4268-4270.

65 Trelea, I.C., The particle swarm optimization algorithm: convergence analysis and parameter selection. *IPL: Inform. Process.Lett.* 2003; **85**(6): 317-325.